INTRUSION DETECTION AND MARKING TRANSACTIONS IN A CLOUD OF DATABASES ENVIRONMENT

Syrine Chatti and Habib Ounelli

Department of Computer Sciences, Faculty of Sciences, El Manar, Tunis

ABSTRACT

The cloud computing is a paradigm for large scale distributed computing that includes several existing technologies. A database management is a collection of programs that enables you to store, modify and extract information from a database. Now, the database has moved to cloud computing, but it introduces at the same time a set of threats that target a cloud of database system. The unification of transaction based application in these environments present also a set of vulnerabilities and threats that target a cloud of database environment. In this context, we propose an intrusion detection and marking transactions for a cloud of database environment.

KEYWORDS

Database system, cloud computing, intrusion detection, marking transaction.

1. INTRODUCTION

The unfolding of a cloud of database system has risen over the world cause of the need of distributed storage and the great volume of data handled by business applications. The unification of transaction based application in these environments present a set of threats that can target a cloud of database environment.

For example, the execution of the malicious sub transactions may be delayed by rendering the wireless nodes unavailable or a fake commit or rollback may be issued by the intruder when the system is compromised. As a consequence to these malicious actions critical data may be lost. Available intrusion detection do not support the particularities of these environments (database, cloud and transaction systems) in their processing either to detect or mark attacks.

To overcome such shortcomings, we propose an appropriate intrusion detection and marking transactions in a cloud of database system. This approach includes the definition of the communication model that is defined in order to ensure the exchange of a set of notification messages. This scheme is based of marking the details of transactions by maintaining a set of valuable information regarding the dependency relationship between these transactions and their security status.

The remaining part of the paper is organized as follow. The section 2, we make a survey of the existing approaches for intrusion detection of database systems. In the next section, we will give an overview of the system architecture. In the section 4, the proposed intrusion detection and marking scheme is proposed. In section 5, a simulation results is proposed in order to depict both

DOI: 10.5121/ijccsa.2016.6502

behavior and efficiency about the suggested scheme following system compromission. Section 6 concludes the paper.

2. RELATED WORK

Several techniques have been proposed to handle different aspects of intrusion detection in a database environment

In [2], the authors present the misuse detection system that uses auditing in order to have sources describing typical behaviour of the users using the database system. The drawback of this approach that is detects only known attacks and is unable to detect new attacks.

In [8], Zhong presents the database intrusion detection based on user query itemsets mining with item constraints. In this approach the intrusion detection uses a profile of normal user stored in database that is appropriate to find unknown attack in the database. The main problem in this approach that the delay of execution is long in addition false negative alarm increased.

In [7], a data mining approach for database intrusion detection is presented. A dependency relationships between data items is determined. If one item is modified then another item refer with it is also modified. The authors determine dependency among data items where data dependency refers to the access correlations among data items. The association rules generate these data dependency. Therefore, transactions that do not follow any of mined data dependency rules are marked as malicious one. The problem of this approach that they process all the attributes at the same significance and equal level which is not always true in real applications.

In database intrusion detection using weighted sequence mining [6], they consider that some attributes are more sensitive to malicious modifications compared to others. Any transaction that does not follow these dependency rules are identified as malicious. The most important problem in this approach is the proper support is identificated and the values are confident.

In intrusion detection of Malicious Activity in RBAC administered databases [1], the authors have presented a role based approach to detect malicious behaviour in role access control. A classifier is used in order to build role profiles and detect abnormal behaviour. The classifier estimates roles for given user than it compares it with the actual role of user. If it's different an alarm is triggered. This approach is good for databases which use role based access control. In addition, it covers insider threat scenario. But, the dependency relationship is not detected in this approach. So, some attacks in the database is not detected.

3. SECTION OVERVIEW

In this section, we will introduce the architecture of the proposed system that includes the cloud database controller (CDC), the participant nodes and cloud database storage (CDS) as presented in Figure 1.

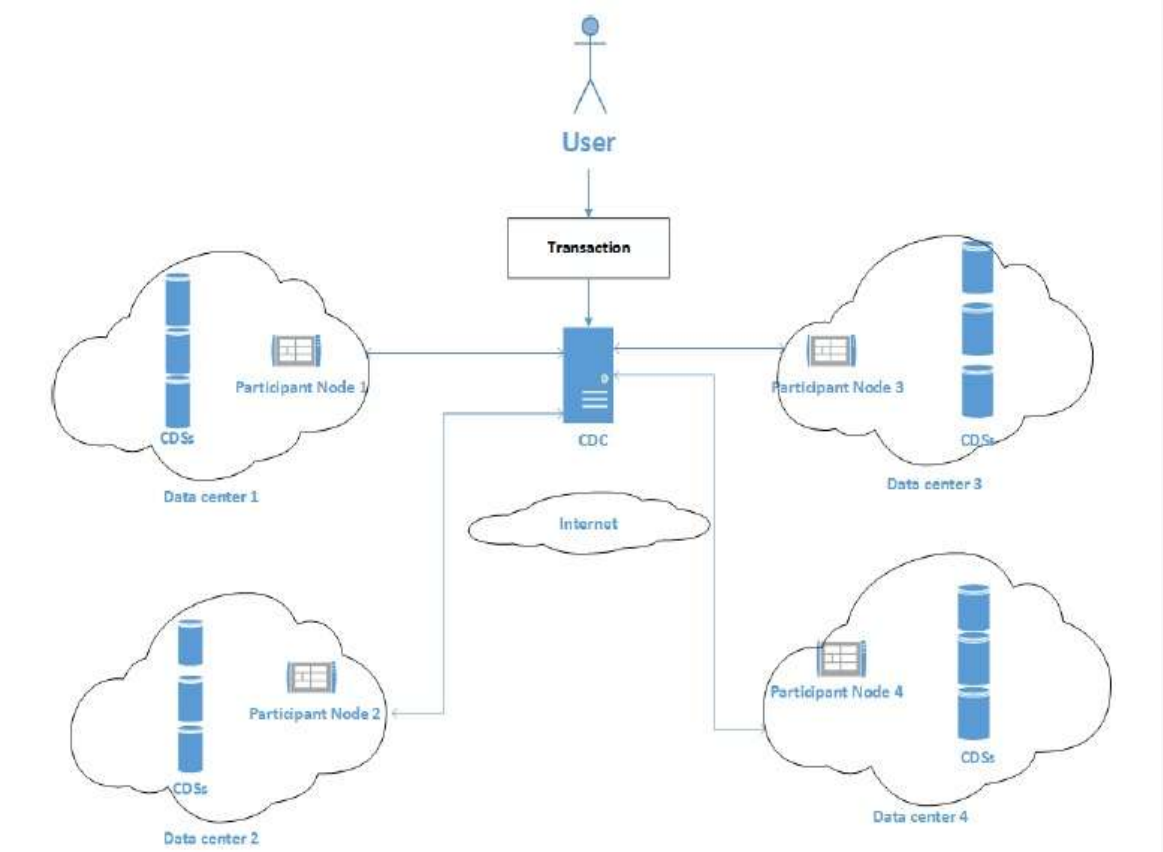

Figure 1 The architecture of a cloud of databases system

- CDC: It's the main component handling the CDS's interactions. A common pool is defined to manage CDSs and CDS blocks are allocated to host system from it. The CDC manages data handled by CDSs and it manages the transactions running in the system. The CDC operates a set of structures to serve incoming requests. It serves and presents a set of structures:
    - The first is a mapping table that hold the transactions and its originator as cited in [3].
    - The second is a table that contains for each input the adequate data center.
    - For each data center, a structure that provides a mapping table that contain the participant node and its suitable CDS.
- Participant Node: The participant nodes are designed for transaction services, for data centers that are located in different geogical locations and corporate data center as well. This link is required for easy and complete access to the database on cloud services.
- CDS: It produces a detailed report on each step used to access data and allowing you to implement precisely improved performance.

3.1. System overview

When receiving the transaction from the user, the CDC divides the transaction into sub transactions, checks its mapping table and then decides which data centers will be the best for processing of the sub transactions.

After identifying the adequate data centers, the sub transactions will be transferred to that specified one. When receiving the sub transactions, the participant node processes it and responds the CDC by issuing a request for a local commit or rollback.

The cloud of database system enables the possibility of storing multiple copies of data at various physical locations throughout the system. Data replication accross multiple sites is desirable for a variety of reasons. To provide disaster tolerance, copies of data stored at different physical location should be available. When a copy becole unvailable because of a default of CDS, breakdown of the network, the replica located at another site can allow access to data. In such architecture, the replica provided benefit until failure.

For the processing of data replication, the CDC have a mapping table that holds for each entry the transaction and its DCS. A strategy of replication is imposed for this purpose, the CDC should replicate the requested data in the destination. Then, it stores copies in different places. Therefore, in its mapping table, the CDC has the data transaction, the CDS and location of replicated data.

3.2. The attacks that target components in our environment

In this section, we will present the attacks targetting the components in a cloud of database system.

1. Against CDC: In order to gain access to the service, a SQL injection attack is realized by the intruder against the authentication server to gather required information during the authentication process. When this step is performed, the intruder utilizes these parameters to subscribes for a session. The CDC authorizes the malicious user to participate in the session. After performing the SDL injection attack, he tests the case, does a SELECT and stocks the user name and passwords. This attack allows showing password and identifiers associated with it. Once a hacker has obtained the password, he modifies an existing SQL commands to expose hidden data so that it can have an access to secret information. After gaining access to the CDC as a super user by using access parameters obtained after the SQL injection attack, the attacker can gain the integrity of files. In fact, when a transaction is initialized, the attacker will try to change the mapping table of the CDC that contains for each entry the transaction and its originator, the server and its associated CDS. In addition to the mapping table, the attacker can change the set of associated blocks in the defined CDS. Therefore, he can attack the integrity of files and modifies the association between the file and the appropriate CDS.

2. Against the participant nodes: The participant node that is responsible for the processing of sub transactions has been compromised because an intruder may define an attack scenario including a transaction and its sub transactions. The execution of the malicious sub transactions may be performed by the intruder by rendering the participant nodes unavailable in order to defeat available intrusion detection systems. For this purpose, an authentication phase between CDC and the participant nodes should be implemented in order to solve this problems depicted in Figure 2. In addition, the participant node may become malicious. In fact, the intruder wil send a malicious file that can have an access to the machine then set up tools and controls the transactions. Thus, a fake commit or rollback may be issued by the intruder when the participant node is compromised. Consequently, critical data may be lost.

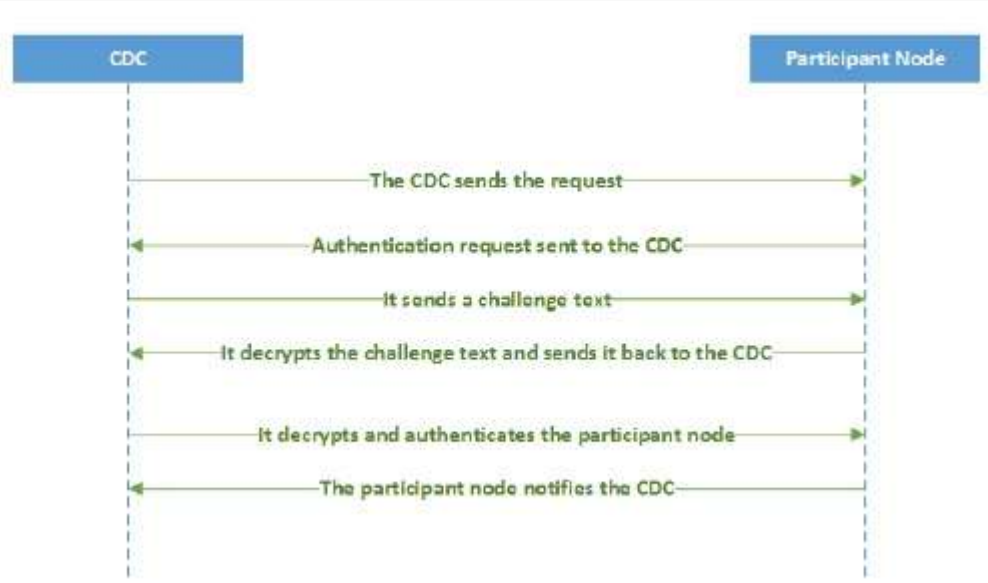

Figure 2. Authentication between the CDC and participant node

4. THE MARKING SCHEME FOR TRANSACTION-BASED APPLICATIONS

This section introduces the marking scheme and gives an overview of the semantics and characteristics of our dependency graph.

4.1The marking scheme for transaction-based applications in WSAN environment

In order to perform marking intrusion in a transaction based WSAN environment, a set of information are collected about running transactions and a generation of dependency graph. The following set of structures and modules, which are depicted in Figure 3 as cited in [3] are described as follows:

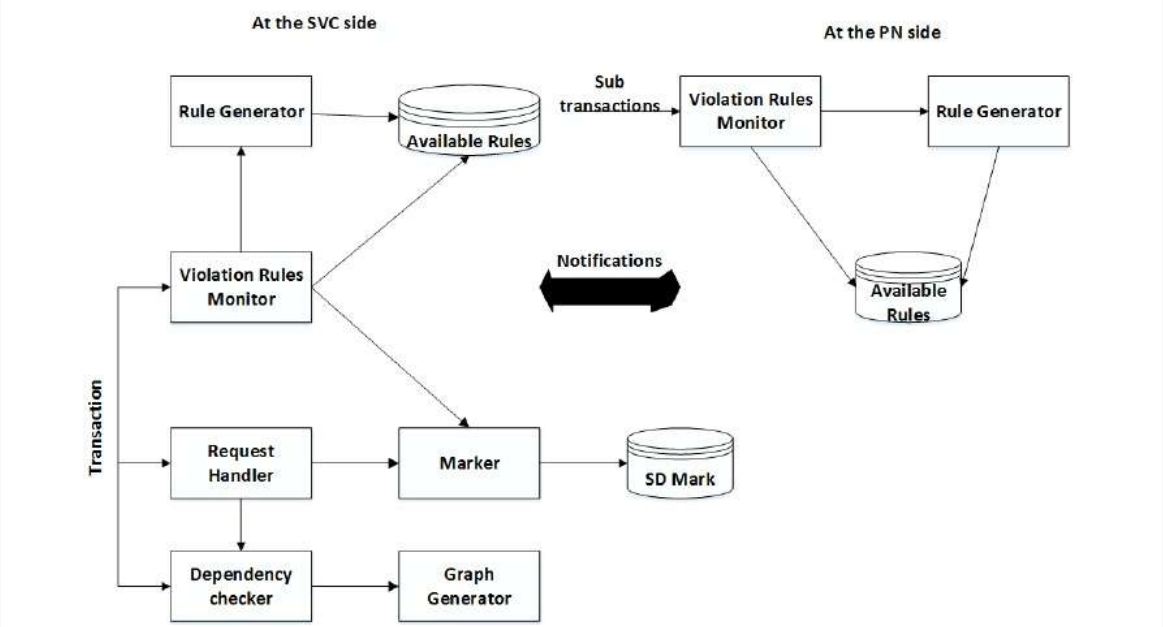

Figure 3 The proposed system architecture

- Structures: In this part we introduce the structures needed for the proposed marking scheme .
    - Transaction's status table: Information that keeps for each transaction the appropriate information about their sub transaction (it's located at the SVC side). Each transaction at this table includes these fields: transaction-id, sub transactionid, processing node-id, decision. The decision field may be legitimate or malicious and the transaction status indicate if it is committed locally or globally or rolled back. Moreover, this table includes entries related to a transaction which its processing has been deleted after a period of time fixed by the SVC.

    - Transaction dependency graph (TDG): The TDG is a directed acyclic graph that is formed by nodes and links. The nodes in the proposed graph does not only represent the transaction identifier but also processing nodes and files. This graph includes in addition to the nodes and transactions two different links: lines between nodes represent parental relationships and dashed links represent the set of requested data in addition to other kind of operations that are performed on them such as read (R) and write (W) operations. Also, we can find the transactions execution schedule.

- Modules: In this section, we will illustrate the modules needed for our proposed scheme .
    - Violation Rules Monitor (VRM): This component is available at both sides: the SVC and the PNs. It monitors incoming request against available rules . At the SVC, the VRM checks the incoming requests against the set of rules then adds its decision in the transaction's status table. At the processing node, the VRM checks the incoming sub transaction against a set of rules, inserts in the mark a set of fields and forwards it to the marker

- Rule Generator (RG): The rule database contains dynamic and static rules. The static rules are initially produced in the database. Otherwise, the dynamic rules are generated after detecting a new malicious activity. During the analysis of the request, the VRM can detect breaches that are transparent to the SVC and PNs sides. Thus, new dynamic rules are produced by the RG.

- Request Handler (RH): It's located at the SVC side, it determines nodes that are needed to process sub transactions and addresses each sub transaction to the adequate PNs for processing. In addition, it affects these information to the transaction's status table. Marker: This component is available at both sides: the SVC and the PNs. Its basic function is the generation of the mark. It identifies the TDG components by collecting the information at the level mark such as the decision, the identifier of transaction and the PN. After performing the decision, the marker inserts in the SD mark the TDG in addition to the mark.

- Dependency Checker (DC): It determines the dependency between the set of running transactions. These dependencies may be an input/output relationships, shared access to data, or execution of several sub transactions by the same PN. These dependencies are written in a specific format that includes several fields such as the identifier of transaction and the type of transaction dependency. Moreover, it analyses a set of transactions in order to return the order of execution of transactions in addition to the order of elementary operations in each transactions.

- The Graph Generator (GG): It generates the TDG by considering the dependency performed by the DC. An example of a transaction dependency graph is described by the Figure 4. Figure 4 illustrates an example of a TDG. Indeed, this graph is composed of 5 nodes. The transaction T1 is the parent of T2 that will be executed after T1 and before T3 which depends on T2 and will be executed on the SD3. Another case is presented in this example: T5 depends on T4 that performs a read execution in SD3 and a write execution in SD2. Therefore, the label w.8.D3 means that the transaction T5 will be executed after the read of T4 in SD3. According to this graph, several rules are generated. For each rule, all malicious combinations related to the processed transactions are marked in the SVC side. According to this format the decision of the SVC is not only limited to the decision of processing nodes but it take into account the dependency between sub transactions and the elementary requests between these running sub transactions. Rules are generated by following a set of steps that starts by analyzing the processed transaction, three rules are generated. For each rule, all malicious combinations should be added to the available detection rules.

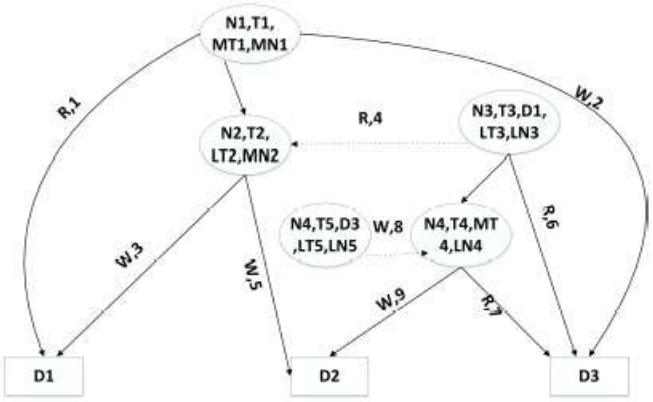

Figure 4. Transaction dependency graph

4.2. Marking scheme

In this section, we illustrate the intrusion marking in addition to the illustration of the mark.

- The mark structure: In order to mark attacks against a transaction-based applications in WSAN environment, a decision of marking is made by the SVC and the PNs based on the content of the transaction dependency graph. In the transaction-based traffic, the marking strategy will be applied during the processing phase for a transaction. When receiving the transaction, the SVC divides the transaction into sub transactions. The RH determines nodes that are needed to process sub transactions and addresses each sub transaction to the adequate processing nodes for processing. Then, it affects these information to transaction's status table by inserting several fields such as: transaction-id, sub transaction-id and the id of processing nodes. When the processing nodes receive sub transactions, the VRM checks the incoming sub transactions against a set of rules. After that, the VRM inserts its decision about the sub transaction and the PN, the id of the sub transaction and the id of the PN in addition to the timestamp in order to prevent the replay attack. Then, it sends a secured notification to the SVC informing its decision. In the other hand, the SVC checks the TDG in order to take a decision about running transactions and updates it. The marker inserts the mark in the SD mark considering the decision of the VRM and the TDG. After the initiation phase, the SVC may have a decision about several sub transactions. Therefore, in order to facilitate the task to the PN, it forwards the decision, the operation and the id of sub transactions that have the relation with the related sub transaction. So, the mark structure will be in this form: ((decision; operation; id transaction); id transaction). In the other hand, we treat the case of shared access to data and level of priority. It can be presented when n transactions will access to the same data with the same or different PNs:

  - With the same PN: The SVC sends to the PN {(operation; id transaction; file; *)}. * means that this chain can be repeated n times and the order of appearance of these chains is the schedule of the transactions execution.
  - With different PNs: The SVC will send {[#]; op; id transaction; file}. # means that the adequate PN will take the decision about the transaction considering the dependency relationships and the decision.

4.3.Communication model

We need the communication model in our proposed scheme for the communication between the SVC and the PNs for sending and receiving notifications. Indeed, when the PN deduces the nature of sub transaction (malicious/legitimate), it sends a notification to the SVC informing it of its decision. In the other hand, after performing its decision, the SVC notifies the processing node about its decision. The notifications sent and received may be attacked by an intruder who can change the decision or the id of the transactions and the related sub transactions. For that purpose, a generation and distribution of encryption keys is presented in both ways. In fact, when the SVC or the PN will send their decision, the SVC or the PN initiates the generation of keys during the notification phase. Then, the SVC or the PN sends it in an encrypted message using the public key extracted from the participant node or SVC certificate

4.4. An intrusion detection and marking transactions in a cloud of database environment

In this section, we will adopt and enhance the idea proposed in the next section in order to detect intrusion and mark transaction.

At the participant node side: When receiving the sub transaction, the participant node checks it against three elements: source, size and encoding.

Source: In order to protect the data, the participant node grants access to the database based on the provenance IP address of each sub transaction. The participant node specify which IP address ranges are allowed. However, if a client wants to electicly grant access to the database, the participant node checks the originating IP address of the request against the full set of rules: if the IP address of the request is within one of the ranges specified in the set of database rules then the connection is allowed to the database. Else if the IP address is not within one of the ranges specified in the set of database rules then the participant node rolled back the sub transaction and notifies the CDC by informing it that the connection is failed as shouwn in Figure 5.

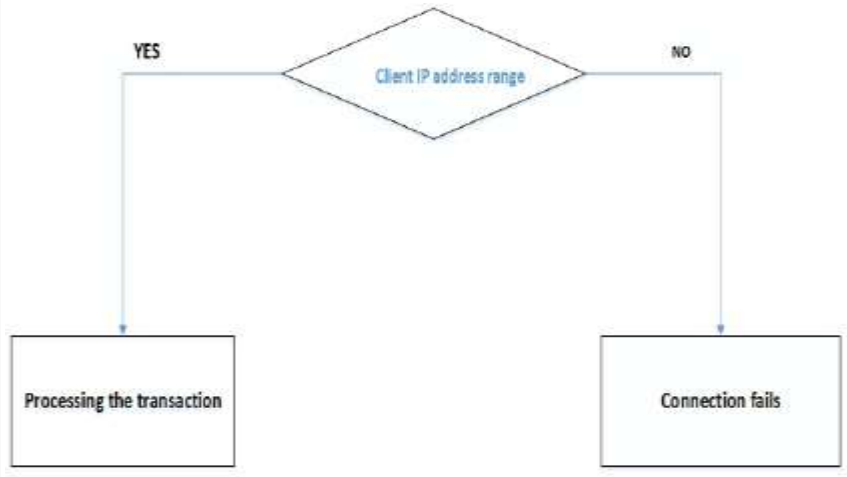

Figure 5. Source identification

Size: When receiving the sub transaction, the participant node may realize that the size of the request will not be supported by the server. Therefore, the participant node will rollback the sub

transaction. A memory thresholds must be defined in order to allow for incoming transactions to manage those that can be described as reliable against these quotas

Encoding: To prevent from the SQL injection, when receiving the sub transaction, the participant node checks it and when the request is encoded, the participant node notifies the CDC informing it that the sub transaction is malicious. For consequence, the participant node rolled back the sub transaction. In the Figure 6, an example of SQL injection attack.

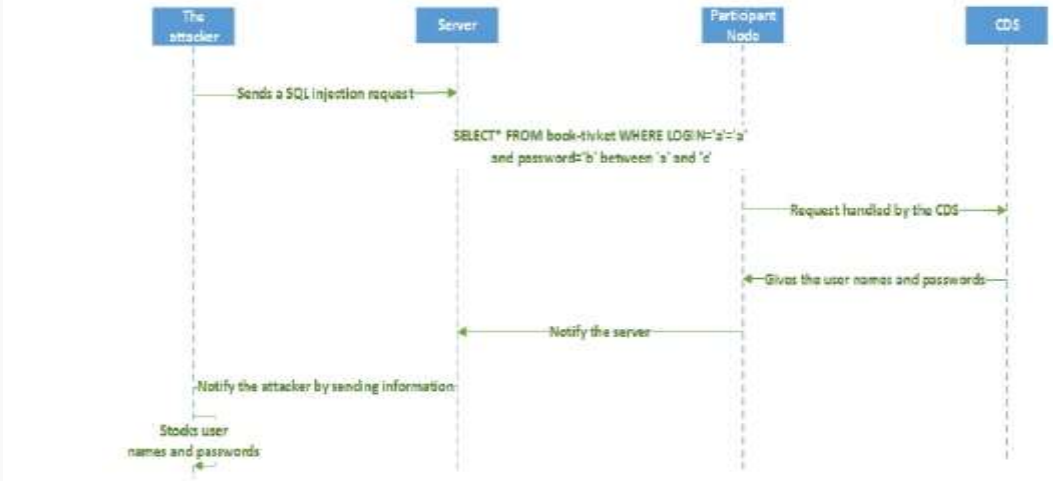

Figure 6. SQL injection attack

At the CDC side: When receiving the transaction, the CDC divises the transaction into sub transactions. The RH determines the data center needed in order to affect each sub transaction to the adequate one. Before forwarding the sub transactions to the data centers, the VRM checks it against a set of rules. After that, the VRM inserts its decision. Two cases are proposed:

- If the sub transaction is malicious then the marker inserts the mark in the SD mark considering the decision of the VRM.
- If the sub transaction is legitimate:

- In Figure 7, a malicious action in sub transaction $t_1$affects in the decision of the CDC. In fact, the VRM checks the sub transaction $t_2$against the set of rules and concludes that is legitimate. Even if its first decision is legitimate, it will be considered as malicious because it depends on t_1, in addition to the elementary request. However if $t_2$depends on t_1, $t_1$is malicious and makes a write operation, and the first decision of the CDC on the sub transaction is legitimate and the elementary operation is read or write operation then the final decision of the CDC is malicious.

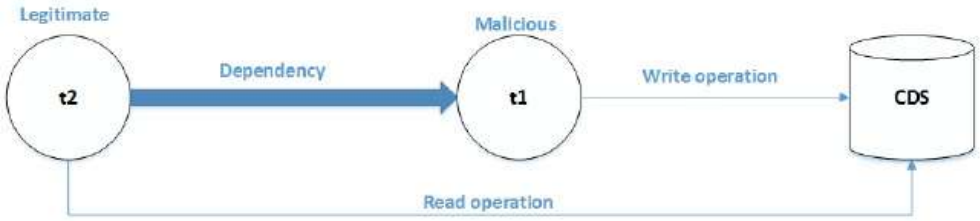

Figure 7. Write operation for dependent transactions

- In this example , we have two sub transactions t_2 and t_3. We assume that t_3 depends on t_2 where t_2 is malicious and makes a read operation. The first decision of the CDC about the sub transaction t_3 is legitimate and the elementary operation is read or write. Therefore, the final decision of the CDC is legitimate because even if t_2 is malicious that does not mean that it affects the CDS as shown in Figure 8.

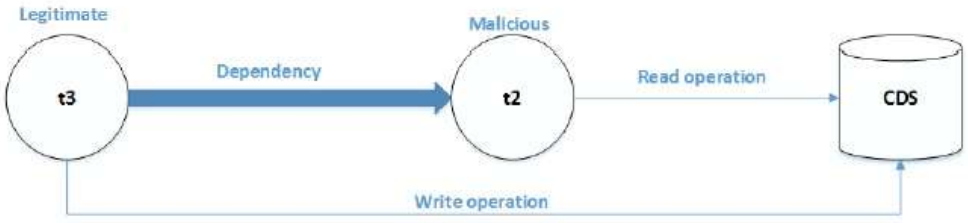

Figure 8. Read operation for dependent transactions

- According to the TDG, the parental relationships exist. Indeed, in the Figure we have $T_1$is parent of $t_2$and t_3. If T_1 is malicious then t_2 and t_3 are malicious. This relationship is available for a transaction and their related sub transactions as described in Figure 9.

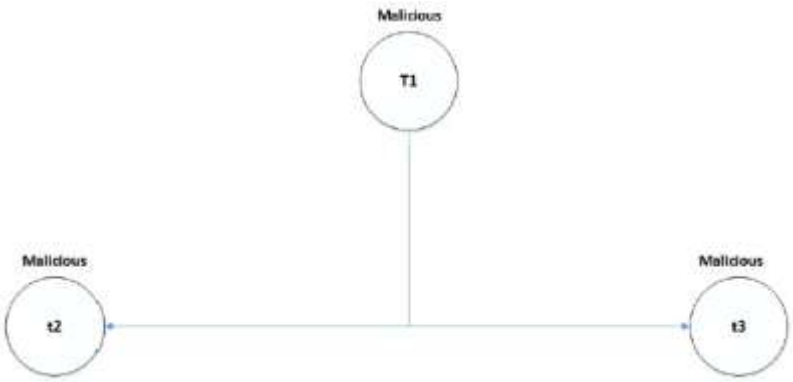

Figure 9. Parent-Child relationships

- After this first phase, the CDC forwards the legitimate sub transactions to the adequate data center. The particioant node will check it against three elements: the source, size and encoding. Then it sends a secured notification to the CDC to inform its decision. We suppose that the decision of the participant node about the sub transaction is malicious and this detecting is a new malicious activity. We know also that the rule database contains dynamic

rules. Then, the RG updates the set of rules by adding these breaches that will be transparent to the CDC side.

4.5. Case Study

This section illustrates the intrusion detection and marking transaction in a cloud of databases environment through a case study that describe the architecture of the proposed marking scheme as shown in Figure 10.

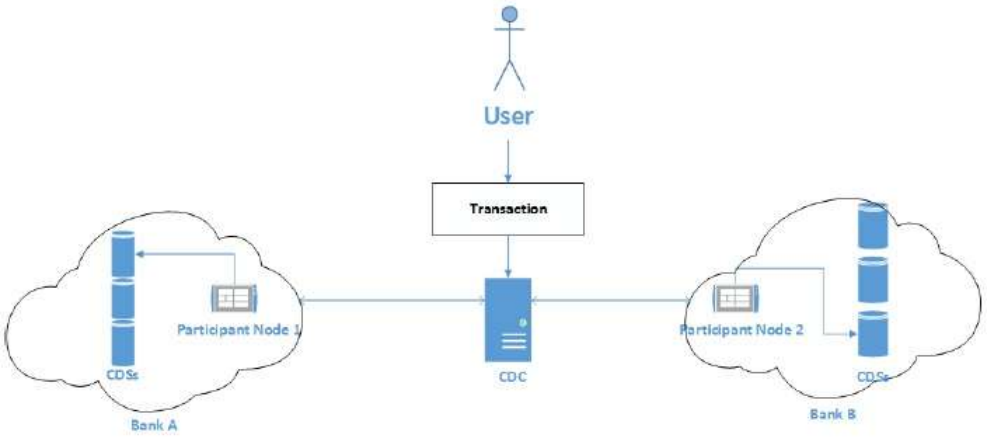

Figure 10. The proposed architecture for CDC and Participant nodes

In order to illustrate the intrusion detection and the marking transaction in a cloud of database environment, we propose a cloud of databases environment which include a set of components that provide service for clients. This system consists of two branches of the bank like described in [4].

A transaction is sent by the CDC to be processed. A client wants to move his account from bank A to bank B. A transaction T is sent to the CDC. The CDC divises the transaction T into two sub transactions $\{t_1, t_2\}$. The RH determines the data center needed .Then, it affects these information to transaction's status table by inserting several fields such as: transaction-id, sub transaction-id and the data center needed. The VRM checks the sub transactions t_1 and t_2 against a set of rules. The dependency checker determines the dependency between the running sub transactions. In our example, t_2 depends on t_1 as shown in Figure 11 because:

t_1 = {o_1 = read old accaount, o_2 = delete old account}
t_2 = {write o_1}

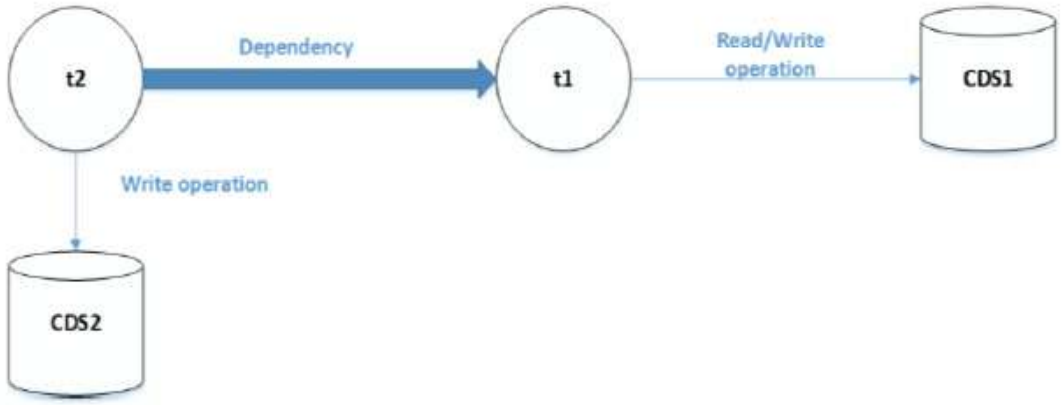

Figure 11. The dependency relationships in the case study

After that, the dependency checker specifies that's an input output dependency. Then, the graph generator generates the TDG by considering the relationships between transactions as presented in Figure 12.

Figure 12. Transaction dependency graph

If t_1 and t_2 are legitimate, the CDC sent the two sub transactions consecutively to the participant node 1 and the participant node 2 in order to check them against the three elements: source, the size and encoding. Therefore, the participant nodes notify the CDC informing it about the decision taken. Otherwise, if t_1 is malicious and makes a write operation that is delete old account then t_2 will be considered as malicious considering the VRM and the TDG. Finally, the marker will mark both of the sub transactions as malicious.

5. SIMULATION OF WSAN AND CLOUD OF DATABASES ENVIRONMENTS

This section illustrates the simulation environment and describes the simulation model that will be evaluated and used to evaluate the efficiency of our marking and intrusion tolerance schemes in WSAN and cloud of databases environments.

5.1Simulation Model

In WSAN environment: In this section, we describe the scenario used to simulate the performance of our marking intrusion scheme. We consider a WSAN environment that integrates an SVC, six PNs and three SDs. When receiving transactions, the SVC divides the transactions into sub transactions and affects each sub transaction to the adequate PNs for processing. Problems of coverage or lack of resources can be presented when transactions are in process. In this case, the SVC initiates a selection phase of new SVC from available nodes. The selection of a new SVC is based on performance criteria including a free processing unit, free memory and a high signal level. Finally, the old SVC will inform the rest of the processing nodes about the assignment of the new SVC.

The SVC must be able to solve these problems in order to ensure and guarantee the processing of running transactions. The objective of our simulations is to prove the efficiency of our approach to detect malicious transactions then to mark it. The conditions could be summarized when we have malicious transactions that affect the whole system without being detected. The simulation scenario includes the definition of dependency between transactions that will be processed by the WSAN components. In addition, the communication model between the SVC, PNs and different distributions needs to be defined in order to illustrate the behavior of the proposed scheme as mentionned in [3].

In a cloud of databases environment: In this section, we describe the scenario used to simulate the performance of our marking and intrusion tolerance scheme.

We consider a cloud of databases environment that integrates an CDC and two data centers that include on their side the participant nodes and the CDCs. The algorithm 1 as given below is used to generate the files containing in the CDC as cited in [5].

```
Map<File, mxCell> storageMap = new HashMap<>();{ CloudDatabaseStorage[] storage = new CloudDatabaseStorage[] { new CloudDatabaseStorage(new File("F1"), new File("F2")), new CloudDatabaseStorage(new File("F3"), new File("F4"), new File("F5")), new CloudDatabaseStorage(new File("F6")), new CloudDatabaseStorage(new File("F7"), new File("F8"), new File("F9")) };for (CloudDatabaseStorage cds : storage)for (File f : cds.getFiles()) storageMap.put(f,(mxCell) transactionGraph.insertVertex(transactionGraphParent, f + "", f, 0, 0, 60, 40)); }
```

Algorithm 1. Generating Files

The TDG generates rules in a way that take into consideration the transaction dependency and the dependency between subtransactions in addition to relationships between the parent and child transactions and relationships between different transactions. In the algorithm 2, the dependency and the parental relationships are described as follow.

```
Map<SubTransaction, mxCell> transactionMap = new LinkedHashMap<>(); Set<SubTransaction> transactionKeys = transactionMap.keySet();{ Document transactionDoc = DocumentBuilderFactory.newInstance().newDocumentBuilder() .parse(new File(args[0])); transactionDoc.getDocumentElement().normalize();NodeList transactionFlow = transactionDoc.getElementsByTagName("SubTransaction");for (int i = 0; i < transactionFlow.getLength(); i++) {Element subTransactionTag = (Element) transactionFlow.item(i);String id = subTransactionTag.getAttribute("id");SubTransaction
```

```
subTransaction = SubTransaction.findSubTransactionById(transactionKeys, id);if (subTransaction == null) subTransaction = new SubTransaction(id);int j = 0;for (NodeList operationsFlow = ((Element) subTransactionTag.getElementsByTagName("Operations").item(0)) .getElementsByTagName("Operation"); j < operationsFlow.getLength(); j++) {Element operationTag = (Element) operationsFlow.item(j);new Operation(subTransaction, operationTag.getAttribute("id"),Operation.Action.valueOf(operationTag.getElementsByTagName ("Action").item(0).getTextContent()),CloudDatabaseStorage.findFileByName(storageMap.keySet (), operationTag.getElementsByTagName("File").item(0).getTextContent())); }Element childrenTag = (Element) subTransactionTag.getElementsByTagName("Children").item(0);if (childrenTag != null) {NodeList children = childrenTag.getElementsByTagName("ChildTransaction");for (j = 0; j < children.getLength(); j++) {id = ((Element) children.item(j)).getAttribute("id");SubTransaction child = SubTransaction.findSubTransactionById(transactionKeys, id);if (child == null) { child = new SubTransaction(id); transactionMap.put(child, null); }subTransaction.addChild(child); } }if (transactionMap.containsKey(subTransaction)) transactionMap.remove(subTransaction);transactionMap.put(subTransaction, (mxCell) transactionGraph.insertVertex(transactionGraphParent, subTransaction + "", subTransaction, 0, 0, 60, 40, "shape=ellipse;perimeter=ellipsePerimeter")); } }
```

Algorithm 2. The dependency and the parental relationships

The objective of our simulation is to prove the efficiency of our approach to ensure the serialization of running transactions.

5.2. Simulation Results

In this section, we aim to evaluate the efficiency of the proposed marking scheme in addition to the intrusion detection capabilities based on the generated marks.

In WSAN environment

In this section, we present our approach to perform the simulation work and obtain the experimental results. We aim to evaluate the efficiency of the proposed marking scheme in addition to the intrusion detection capabilities as described in [3].

The intrusion detection capabilities for the proposed scheme: The detected malicious transactions depend on the available detection rules, the complexity of the dependency scheme between sub transactions in addition to the number of transactions to be processed by the WSAN components. Indeed, the available detection rules may increase with the dependency relationship of sub transactions and as a result, the number of detected malicious transaction increases. In this simulation, we have considered two types of detection rules: static and dynamic. Indeed, in the static mode, the available detection rules are a set of rules that specify a set of malicious actions on the defined resources during the simulation. Otherwise, the dynamic available rules may be attached to new added resources or related to actions on dependent transactions. Therefore, dynamic rules are generated during the simulation by the increase of the number of transactions in addition to the dependency between sub transactions. Figure 13 illustrates the variation of the number of malicious transactions detected in function of the total number of the malicious transactions that are operated in the system, including the legitimate transactions and the malicious ones in addition to the dependency relationships between them. Moreover, Figure 13 shows the comparison between the number of detected malicious transactions when using dynamic available rules and the number of the detected malicious transactions with static

available rules. Indeed, the first curve represents the number of malicious transactions among all the transactions that are operated in the system. In the other hand, the second curve and the third curve represent the number of the detected malicious transactions compared to the total number of malicious ones. Moreover, the number of the detected malicious transactions depends on the type of available rules, which is either dynamic or static. As depicted in Figure 13, 30 malicious transactions are detected using available dynamic rules while the number of malicious transactions detected with static rules are 24. Also, when we have 10 transactions, we can detect just 2 malicious transactions with static rules. Otherwise, 4 malicious transactions with dynamic rules. Therefore, this observation illustrates the effectiveness of the dynamic rules regarding the detection of malicious transactions since the number of the detected malicious transactions is more important than the one obtained with static rules.

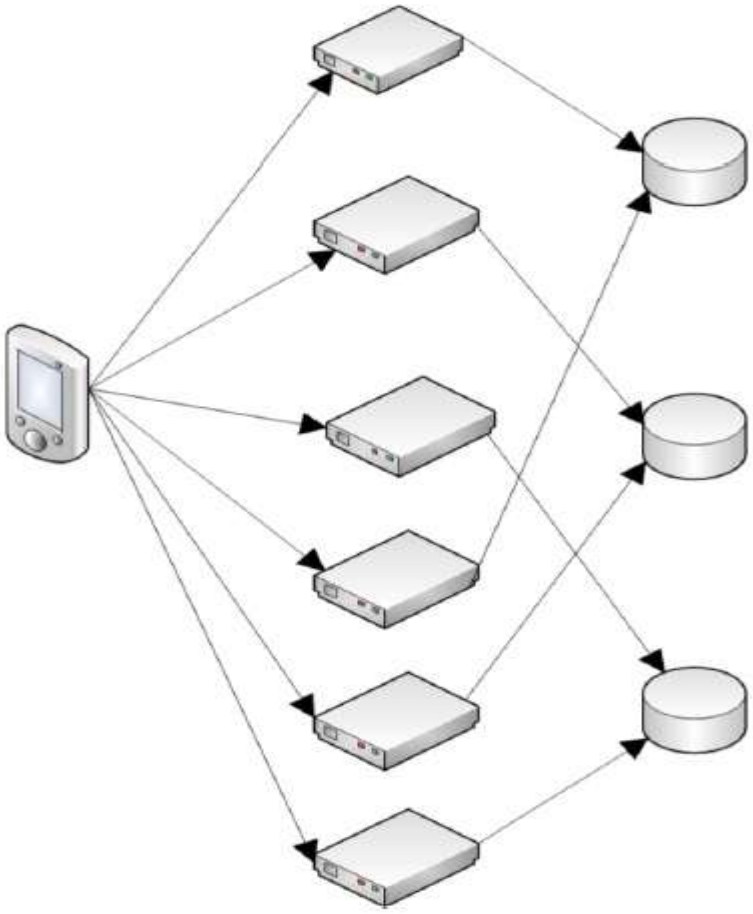

Figure 13. Malicious transactions detected compared to the total of malicious transactions

Evaluation of the intrusion detection capabilities based on the marking scheme: In this simulation, we study the malicious transactions detected with marking support compared to the malicious transactions detected without marking. When receiving the transaction, the SVC checks it against a set of rules. After that, it verifies the dependency between the transactions in order to take a decision about running transactions. Then, the SVC inserts its decision and forwards each sub transaction to the adequate PNs. When the PNs receive sub transactions, they check them against a set of rules. Upon the verification, the PNs insert their decision about the verified sub

transactions as well as both ids of the considered sub transaction and PN. Then, a notification is sent to the SVC informing the PN's decision. In this simulation, we can remark that the use of marking scheme can increase the number of detected intrusions. The number of malicious transactions detected with marking support are greater than the number of malicious transactions detected without marking support when using the dynamic detection rules. Therefore, we can conclude that the marking support enhances the detection capabilities of the proposed scheme. In our simulation, for 36 malicious transactions, we have detected 32 malicious transactions with marking support and 30 without marking. Figure 14 illustrates the efficiency of our proposed marking concept in detecting malicious transactions. Indeed, Figure 14 presents three curves: Firstly, the grey curve shows the total number of malicious transactions. Secondly, the orange curve presents the number of malicious transactions detected without marking support. Therefore, comparing with the first curve, we can notice that the ratio between the total number of malicious transactions and the detected malicious transaction is important. Finally, the blue curve presents the malicious transactions detected with marking support. In fact, the blue curve approaches to the orange one and consequently, we can approve the efficiency of the marking approach.

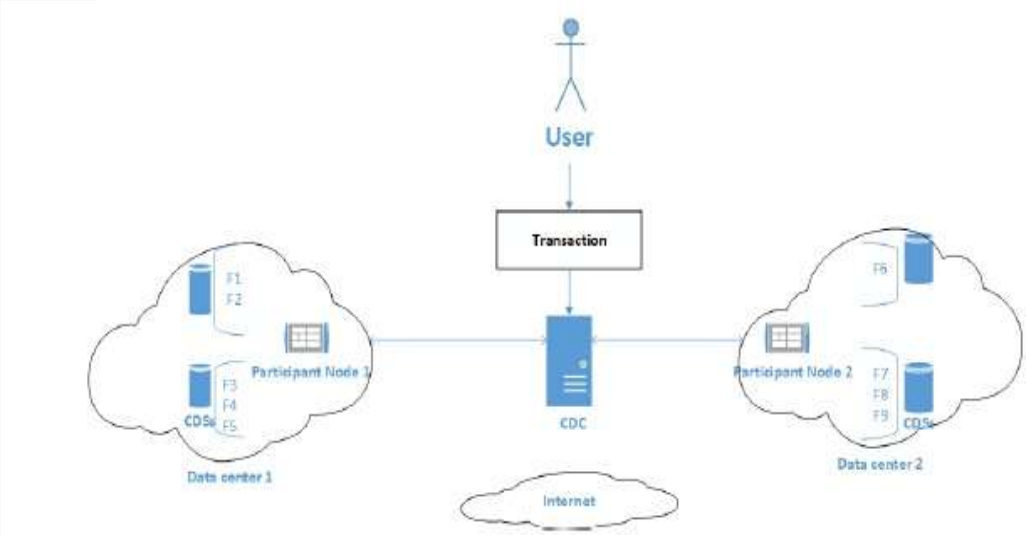

Figure 14. Detection capabilities with/ without marking

In a cloud of databases environment:

The transaction dependency graph: In this simulation we have presented a transaction dependency graph in Figure 15.

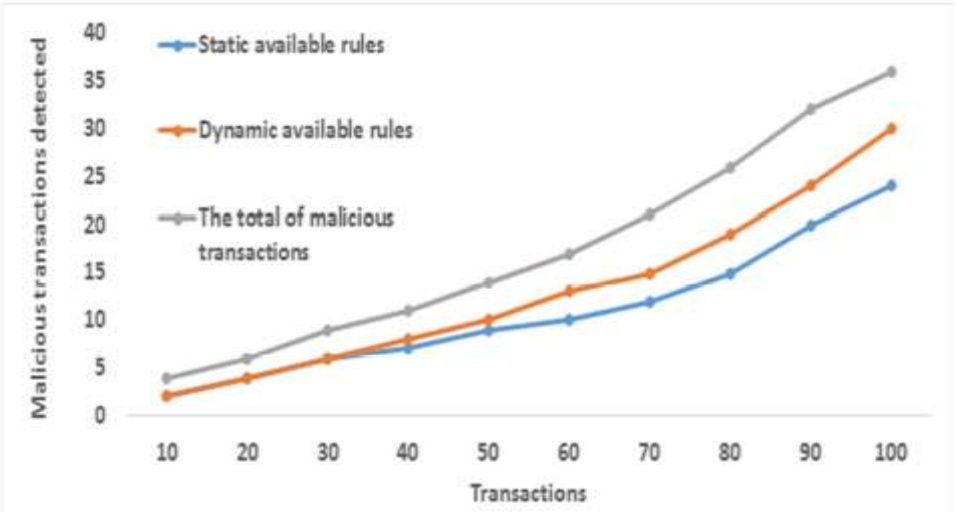

Figure 15. Transaction dependency graph

*T*7 will do a read on a file 2 after the write of the subtransaction *T*6 on file 2. The TDG generates rules in a The Figure 15 illustrates the simulation result of the TDG. Indeed, this graph is composed of seven nodes that describe the subtransactions. The subtransaction*T*1 is the parent *T*2 that will be executed after *T*1 and before *T*3 which depends on *T*2 and will be executed on file 6. Another case can be presented is *T*7 depends on *T*6 has a write execution in file 2 and a read execution in file 7. Therefore the label R,12 means that the subtransaction way that take into consideration the transaction dependency and the dependency between subtransactions in addition to relationships between the parent and child transactions and relationships between different transactions [5].

Marking intrusion: A simulation result is depicted in Figure 16. In Figure 16, the subtransactions *T*3 to *T*7 are colored in red. In fact, the subtransactions *T*3 and *T*7 are the first malicious subtransactions according to the set of rules. Then, with the help of the TDG the other subtransactions became also malicious as depicted in Figure 16 and described in the Algrorithm 3. In the Figure 16, *T*3 is the parent of *T*4 and *T*6, *T*3 is malicious then *T*4 and *T*6 will become malicious. *T*5 will read the file 8 updated by *T*4. Therefore, the subtransaction *T*5 will become malicious.

Figure 16. Marking intrusion

```
public void markAsMalicious(SubTransaction[] subTransactions) {malicious = true;for (SubTransaction child : children) child.malicious = true;int i = 0; while (i < subTransactions.length && subTransactions[i++] != this) ;for (int j = i + 1; j < subTransactions.length; j++) for (Operation nextOp : subTransactions[j].operations)for (Operation op : operations)if (nextOp.getTarget() == op.getTarget()) subTransactions[j].markAsMalicious(subTransactions); }public boolean searchMaliciousDependency(SubTransaction[] subTransactions) {for (int i = 0; i < subTransactions.length && subTransactions[i] != this; i++)if (subTransactions[i].malicious) for (Operation prevOp : subTransactions[i].operations)for (Operation op : operations)if (prevOp.getTarget() == op.getTarget()) return true;return false; }
```

Algorithm 3. Intrusion detection and marking transactions

6. CONCLUSION

We have proposed a marking scheme of Transaction-based Applications in Wireless Storage Area Networks that is mainly based on a transaction dependency graph. We have adopted and enhanced this scheme in order to detect intrusion and mark transactions in a cloud of databases environment. The proposed scheme introduces the intrusion detection and marking transactions by using at first the rules set and the TDG. Secondly, the use of the elements at the side of the participant nodes. The enhancement was by including additional elements related to the processing environment in the cloud of databases. For future research, we project to find the identity of the user. Moreover, we enhance the protection of the mark. Another perspective to this work consists in studying the intrusion tolerance with marking support for interconnected databases.

REFERENCES

[1] Kamra A Terzi E Bertino E, Vakali A. Intrusion Detection in RBAC-Administered Database. In Proceeding of the 21st annual computer security application conference (ACSAC), pages 170-182, 2005.
[2] M. Gertz C. Y. Chung, K. Levitt. DEMIDS: A Misuse Detection System for Database Systems. In Proceedings of the Integrity and Internal Control in Information System, Pages 159-178, 1999.
[3] S.Chatti, H.ounelli. Marking Itrusion of transaction-based applications in WSAN . Network-Based Information Systems (NBiS), 16th International Conference on, 2013.
[4] S.Chatti, H.Ounelli. Transaction Management in WSAN Cloud Environment. 17th International Conference on Network Based Information Systems (NBiS), Pages 525-530, 2014.
[5] S.Chatti, H.ounelli. An Intrusion Tolerance Scheme for a Cloud of Dtatabases environment. 19th International Conference on Network-Based Information Systems (NBiS), 2016.
[6] Sural S Srivastava A, Majumdar AK. Database Intrusion Detection Using Weighted Sequence Mining. Journal of Computers, vol. 1, no. 4, 2006.
[7] B. Panda Y. Hu. A data mining approach for database intrusion detection. In Proceedings of the ACM Symposium on applied computing, pages 711-716, 2004.
[8] Qin X Zhong Y. Database intrusion detection based on user query frequent item sets mining with constraints. In Proceeding of the third international conference on information security, pages224-225, 2004.

Authors
Syrine Chatti : PH.D student in Faculty of Sciences of Tunis
Habib Ounelli: Professor in Faculty of Sciences of Tunis