An Immunity-based Error Containment Algorithm for Database Intrusion Response Systems

Nacim YANES[1], Ayman M. MOSTAFA[2], Nasser ALSHAMMARI[3], Saad A. ALANAZI[4]
College of Computer and Information Sciences–Jouf University–KSA[1, 2, 3, 4]
RIADI Laboratory–La Manouba University–Tunisia[1]
FCI–Zagazig University–Egypt[2]

***Abstract*—The immune system has received a special attention as a potential source of inspiration for innovative approaches to solve database security issues and build artificial immune systems. Database security issues need to be correctly identified to ensure that suitable responses are taken. This paper proposes an immunity-based error containment algorithm for providing optimum response in detected intrusions. The objective of the proposed algorithm is to reduce the false positive alarms to the minimum since not all the incidents are malicious in nature. The proposed algorithm is based on apoptotic and necrotic signals which are parts of the immunity structure in human immune system. Apoptotic signals define low-level alerts that could result from legitimate users but could be also the prerequisites for an attack, while necrotic signals define high-level alerts that result from actual successful attacks.**

***Keywords*—Database security; artificial immune system; error containment algorithm; database auditing; apoptotic signal; necrotic signal; secret sharing**

I. INTRODUCTION

Artificial immune system (AIS) is a field of research that links different disciplines such as immunology, computer science and engineering [1] [2]. AIS is considered the artificial simulation of natural immune system (NIS). The immune system is responsible for guarding the human body against the foreign and dangerous microorganisms called pathogens. To overcome these pathogens, the immune system depends on innate and adaptive immune subsystems [3]. The innate immune subsystem is considered the immutable first line of defense for alarming danger signals around suspicious item. The adaptive immune subsystem relies on a faster response to unknown detected patterns.

Over the last two decades, a rich set of biological immune inspired algorithms have been developed to solve computational problems. Different researches of AISs have been developed and applied based on immune modeling, theoretical artificial immune system, and applied AISs [4]. Works on immune modeling cover several models based on natural immune systems, while theoretical artificial immune system discusses the theoretical field of artificial immune system.

The most active field in AIS is the development of immune-inspired algorithms to apply artificial immune system in diverse real world applications. Negative selection algorithms, clonal selection algorithms, artificial immune networks, and danger theory; are four major algorithms of artificial immune system that are applied in different domains such as intrusion detection systems, neural networks, and data analysis [2, 4].

Protecting the privacy and integrity of data with maintaining high detection rate with low false positive and false negative alarms is a challenging issue for database security. As presented in our previous work [2], the immunity-based detection algorithms are used to protect database from malicious users or intruders who may abuse their privileges to produce hostile acts. The developed detection algorithms produce efficient results in preventing attacks from breaching the security system but the security system may still have vulnerabilities. However, there are two vulnerability cases into which the intruder may attack the immunity-based detection algorithms. First, if the malicious user succeeds in obtaining some confidential information about the privileges of authorized users, he can predict the authentic factor of the authorized user. Second, the malicious user can perform a brute-force attack on the authentic factor set of an authorized user until a valid factor is obtained or the authentic factor set which is considered as a detector of the system fails in detecting the intruder. In both cases, the malicious user can breach the system and obtain confidential information from database. That's why an error containment strategy is crucial and helps identify post-security issues and resolve them quickly.

The main goal of this paper is to propose an error containment strategy with the objective to reduce the false positive alarms to the minimum. The proposed strategy is biological immune inspired; it is based on apoptotic and necrotic signals that are parts of the immunity structure in human immune system. Apoptotic signals define low-level alerts that could result from legitimate or illegitimate users, while necrotic signals define high-level alerts that result from actual successful attacks. Database auditing mechanisms will be embedded in the proposed error containment strategy in order to monitor and audit all user transactions.

The body of this paper is structured as follows. A review of the related works is presented in Section 2 whereas the proposed immunity-based error containment strategy is detailed in Section 3. Section 4 deals with the implementation of most common mechanisms for database auditing. Finally, the paper ends with a conclusion and future works.

II. Related Works

Artificial immune system covers different models and algorithms inspired by biological immune systems [1].

The main mechanism of artificial immune system (AIS) is based on detecting computer viruses, intruders, or threats by generating a set of detectors whose role is to defend application environments. The detectors are used to scan the developed applications and if there is a matching between the detector and any intrusion, the intrusion will be blocked.

One of the researches of artificial immune system is presented in [5]. In this research, a cryptographic algorithm was developed based on the inspiration of artificial immune systems. The developed cryptographic algorithm uses the advanced encryption standard (AES) in order to generate a random output which cannot be predicted by intruders.

The developed algorithm is based on the interaction between antigen and antibody. The key and plaintext are represented as antigen and antibody respectively. A lockup table is developed based on a combination between the key and plaintext for generating the cipher-text. The process of generating cipher-text is executed in 10 rounds to produce random output.

A survey on artificial immune system as an inspiration for anomaly based intrusion detection systems has been presented in [6]. In this research, a set of unique features of human immune systems has been presented such as: dynamic, distributed, diverse, parallel management, self-learning, self-adaptation and self-organization. The features of human immune system can encourage researches to simulate these features in the artificial immune system to provide wide applications. This research focuses also on intrusion detection systems (IDS) using artificial immune systems (AIS).

Another adaptive intrusion tolerance strategy in light of artificial immune systems has been presented in [7]. In this research, the authors introduced two approaches in the intrusion tolerance system: attack response and attack mask.

In the attack response sub-system, when an attack is detected, the reaction time is activated and all system resources are reallocated to continue working normally under attack. In the attack mask sub-system, the overall system will mask the affected part by redundancy, and majority voting.

The first method is simple and has low cost because the structure of the original system is not changed while the second method redesigns the whole system using artificial immune technology, and redundant technology for cloning the system resources. As a result, the method cost will be high.

An important model for the classification of heterogeneous data with artificial immune system is presented in [8]. The transformation of data from their original form into any other specific types cannot only reduce its originality but it can also occupy more space and require more preprocessing time. This model is able to process data with any type without resorting to data transformation.

A new framework for access control in light of the immune mechanism is presented in [9]. In this article, the framework of access control comprises the following: subject, object, access control decision facility (ACDF), access control enforcement facility (ACEF), as well as access control information / access control rule (ACI / ACR). Subject which is the user of certain processes sends out the access. Object is the program, process, data, and information. Access control decision facility (ACDF) enables the subject to visit the object according to the access control rule (ACR) and the access control information (ACI) and provides the result to ACEF. Access control enforcement facility (ACEF) governs the access of the subject to the object. As for Access control information / access control rule (ACI / ACR), it is used to refer when ADF conducts the decision-making, perhaps being deposited in the database, the data file, and chooses other access methods, in light of the security sensitivity and the access control information quantity.

Another intrusion detection system for computer networks based on artificial immune systems is presented in [10]. In this research, a set of randomly generated binary strings that represent the detectors are trained to draw a distinction between the self and non-self connections. When the detector has been provided to all the self and non-self connections, it forms the “Mature Detector Set” and is not subject to further change. This is considered as one of the limitations in this research. This research assumes that the detectors are complete and cannot be changed. This means that if any antigen (intruder) succeeds in passing from the mature detectors, the system will not be able to modify the mature detectors to be capable for detecting the unknown intruder. Another limitation in this research arises during the training of the detector set. The non-self is classified as a hole when a non-self (anomaly) cannot match 3 detectors or more. The authors overcome the problem of holes by developing a solution that randomly generates detectors until the anomaly matches not less than 3 detectors. This solution will cause a great space complexity because the system will generate large number of detectors to detect intruders.

Chen et al. [11] developed an immunity-based intrusion detection proposal for database systems. The developed model provides each detector with an age and an alarm to automatically detect malicious transactions. The Immune System is in charge of producing as well as managing the immunocytes for possible malicious transactions.

Most artificial immune database security researches focus on detecting intrusions that breaches database authentication mechanisms. Although the proposed detection algorithms produce efficient results in preventing attacks from breaching the security system, the security system may still have vulnerabilities. An additional layer of security can be added by applying database auditing that is considered a crucial mechanism for post-security countermeasures.

Database auditing is the mechanism for monitoring user behavior in database. By implementing access control policies, authentication, and other cryptographic techniques, the security of database can be elevated.

Different frameworks are presented to audit database systems. One of the recent security frameworks was presented in [12]. This framework is based on an auditing strategy

management for configuring database authorizations and alarms. Several event actions are applied to track all transactions such as event manager, event generator, event collector, event reporter, event analyzer and event memorizer.

Another auditing framework was introduced in [13] to avoid database performance delays. This framework is used by applying a three-way handshake of TCP data flows. A hash table is used to manage connections for new data packets.

Database auditing can be also used to secure statistical database [14]. In statistical database, users can acquire statistical queries like (average, sum, count, etc.) but specific individuals' information should remain confidential. The aim of the key representation auditing scheme is to guard online and dynamic SDBs from disclosure. The idea relies on the conversion of the original database D into key representation database D|. Thus, before being stored in the Audit Query table (AQ table), each new user query q would be converted from string into key representation query q|.

Another auditing method for auditing mathematical statistics was presented in [15]. In that research, a statistical analysis was conducted to analyze user behavior based normal records. The method is based on memory mathematical statistics to store auditing objects into the memory to analyze user behavior. The authors in [16] present a logging scheme for database auditing used for analyzing and monitoring network traffic. The architecture of this scheme comprises three primary parts: packets capturing and parsing, as well as data storage. First the packets are captured to and from the database. After database communication protocols are analyzed, the captured packets are parsed and immediately used to support database audit.

Auditing the changes to a database is important for improving system performance, maintaining data quality and detecting malicious behaviors. However, an accurate audit log is a historical record that constitutes a serious threat to privacy. The policies that limit data retention clash with the purpose of accurate auditing. Thus, data owners should carefully assess the need for these policies in compliance with the accurate auditing goals. The authors in [17] develop a framework for auditing the changes to a database system while the data retention policies are still respected. The framework consists of a historical data model that supports flexible audit queries, besides a language for retention policies that conceal individual attribute values or delete entire tuples from history. The audit history is partially incomplete under retention policies. The interpretation of audit queries on the protected history is formalized and may contain imprecise results. Policy application and query answering are efficiently implemented in a standard relational system, and characterize the cases where the achievement of accurate auditing under retention restrictions is possible.

III. The Immunity-based Error Containment Strategy

The current section explains different mechanisms to secure database user transactions based on three interactive sequential processes starting with detecting malicious intruders using our proposed immunity-based detection algorithm that was published in [2]. The next process presents our proposed error-containment algorithm. The final process explores a system hibernation framework for auditing user transactions whether to be granted or denied.

A. The Intruder Detection Algorithm

In our paper [2], we proposed an immunity-based detection algorithm to protect database from malicious users or intruders who may abuse their privileges to produce hostile acts. As presented in Fig. 1, the proposed intruder detection algorithm is based on five nested stages. These stages are presented as follows:

1) Stage 1: Verified factor authentication (VFA): The user must pass his/her 18 bits in a correct manner so as not to be detected as a malicious user. If the user passes his/her authentic factor, he/she can move to the last layer of security which is user certificate authorization (UCA); otherwise the security system will perform a set of serial checking mechanisms which are factor length matching, antigen table matching, RCB matching, and DVS matching.

2) Stage 2: factor length matching (FLM): If the user fails in verifying his authentication factor, the first checking mechanism of the intruder detection algorithm is to match the privilege factor length with the user entry. If the factor length is not correct, the security system will raise the danger signal II alarm. Otherwise, the system will proceed to the next checking mechanism which is antigen table matching.

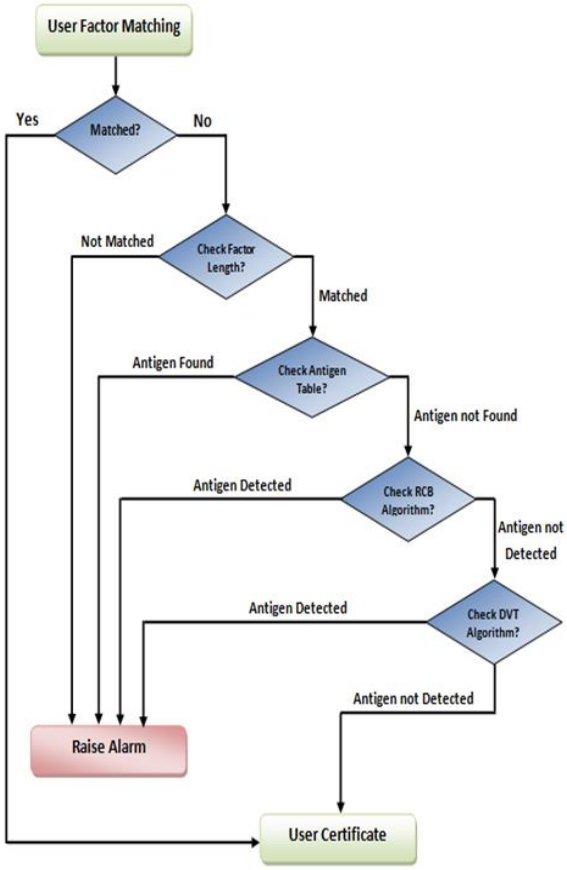

Fig. 1. Intruder Detection Checking Mechanisms.

3) Stage 3: antigen table matching (ATM): The developed antigen table is the learning and memorization stage that stores all previously detected users for performing fast detection response to unknown patterns. The security system searches the user factor in the antigen table to get a quick response instead of applying the detection algorithm.

4) R-Contiguous bit matching: The security system traces each bit of the user factor and matches it with the authentic user factor stored in the "system cache". As presented in [2], the security system will raise danger signal II alarm if at least R-contiguous bits are matched in both authentic and fake factors. Otherwise, the security system will activate the second detection algorithm which is the danger value algorithm (DVS).

5) Stage 5: danger value signal (DVS) algorithm: The last detection stage in the proposed intrusion detection algorithm is to initialize the danger value signal (DVS) algorithm. The main idea of this algorithm is to detect unauthorized users who succeed in passing the four previous detection mechanisms.

B. The Error Containment Algorithm

The intrusion detection algorithm, described in the previous section, produces efficient results in preventing attacks from breaching the security system but the security system may still have vulnerabilities. In fact, the intruder may attack the immunity-based detection algorithms into two vulnerability cases. First: if the malicious user succeeds in obtaining some confidential information about the privileges of authorized users, he can predict the authentic factor of the authorized user. Second: The malicious user can perform a brute-force attack on the authentic factor set of an authorized user until a valid factor is obtained or the authentic factor set which is considered as a detector of the system fails in detecting the intruder. In both cases, the malicious user can breach the system and obtain confidential information from database.

Algorithm 1 presents an error containment strategy for preventing malicious users from harming the system if they succeed in breaching the detection algorithms.

Algorithm 1: Apoptotic-Necrotic Signal Algorithm

```
1.  Integer m=Apoptotic Signal
2.  Integer n=Necrotic Signal
3.  Get User Factor from System cache
4.  Set Detector = User Factor
5.  If strFactor = Cached Detector Then
6.   {
7.      Check User Certificate Authorization (UCA)
8.  If UCA = TrueThen
9.       {
10.    Assign Authentic User
11.    Assign Predefined Privileges
12.    Transactions Committed
13.  }
14. Else
15.   {
16.     Raise Danger Signal III Alarm
17.     User Disconnect
18.   }
19.  }
20. Else
21. {
22. Check RCB Algorithm   // Algorithm 5.2
23.    Check DVS Algorithm   // Algorithm 5.3
24. If strFactor = Detected Then
25.      {
26. Raise Danger Signal II Alarm
27. User Disconnect
28.       }
29.    Else
30.      {
31. Assign Suspicious User
32.          Classify Predefined Privileges
33. For Privileges Between 1 and m Loop
34.           {
35. Assign Apoptotic Signal
36. Transactions Pending
37.              System Hibernation
38.              Database Auditing
39.              Send DBA Broadcasting Request
40.              If Request = Approve Then
41.                 Transactions granted
42.             Else
43.                 Transactions denied
44.           }
45. For Privileges Between m+1 and n-1 Loop
46.           {
47. Assign Necrotic Signal
48. Transactions Pending
49.              System Hibernation
50.              Database Auditing
51.              Send DBA Broadcasting Request
52.              If Request = Approve Then
53.                 Transactions granted
54.             Else
55.                 Transactions denied
56.           }
57. For Privileges = n Then
58.           {
59. Assign Max Necrotic Signal
60. Transactions denied
61.              User Disconnect
62.           }
63.      }
64. }
```

As described in Algorithm 1, the strategy begins by defining apoptotic and necrotic signals. Apoptotic and necrotic signals are part of the immunity structure in human immune system [1]. Apoptotic signals define low-level alerts that could be issued by legitimate users or as a sign of a preliminary attack. Necrotic signals define high-level alerts that result from actual successful attacks. Defining the apoptotic and

necrotic signals are the responsibility of the super administrator (SA). The super administrator (SA) defines the apoptotic and necrotic signals to be used as countermeasure for preventing unauthorized users from disclosing confidential information from databases if they succeed in breaching the security system.

When a user connects to the security system, he must pass the intruder recognition system as presented in [2]. If the user passes the intruder recognition system with a valid username and password, then the user exceeds the first danger signal (Danger Signal I). As a result, the system verifies that the user could be an authorized user. Therefore, the security system retrieves the authentic factor set (AFS) of the user who owns the username and password. The security system retrieves the authentic factor set (AFS) from the system cache and this factor is kept secret until the user verifies his/her identity by passing the valid authentic factor set. The authentic factor set (AFS) is promoted to be the detector if the user fails in verifying his identity.

If the user factor matched the authentic factor set (AFS), then the user exceeds the second danger signal (Danger Signal II). The security system will pass the user to the final verification signal which is the user certificate authorization (UCA). If the user certificate authorization is authentic, the user will be a normal user and can use his/her predefined privileges in passing database transactions via the database server. These transactions will be committed in the database server as they have been executed from normal users. If the user certificate authorization is not authentic, then danger signal III alarm will be raised and the user will be disconnected from the security system.

If the user factor did not match the authentic factor set (AFS), the security system will activate the RCB and DVS algorithms to be used as detection algorithms to detect the malicious user. If the detector succeeds in detecting the malicious user, danger signal II alarm will be activated and the user will be disconnected from the security system.

If the detector failed in detecting the malicious user, the security system will assign the user as “suspicious user” and will pass the user to the final verification signal, which is the user certificate authorization (UCA). The suspicious user may perform a brute-force attack until a valid user certificate authorization is obtained. The apoptotic and necrotic signals which have been developed by the super administrator (SA) will be activated to limit the authorizations of the user.

Apoptotic and necrotic signals activation depends on the probability of three conditions. First: the malicious user succeeds in obtaining a valid username and password and passed the danger signal I alarm. Second: the malicious user succeeds in breaching the detection algorithms and passed the danger signal II alarm. Third: the malicious user succeeds in passing the danger signal III by performing a brute-force attack on the user certificate authorization until a valid certificate is obtained.

The super administrator (SA) defines the apoptotic signal by determining a number of privileges from 1 to m where m is the number of transactions allowed for the suspicious user to perform on the database server. The super administrator (SA) defines also the necrotic signal by determining a number of privileges from m+1 to n-1 where n is the maximum number of transactions allowed for the suspicious user to perform on the database server.

If the suspicious user passed the three danger signals, he can perform different transactions on database until the number of transactions equal m. At this point, apoptotic signal is raised and all transactions are suspended. If the suspicious user performed other transactions, the number of transactions will be incremented until the transactions equal n-1. At this point, necrotic signal is raised and all transactions are suspended. The Database administrator performs auditing mechanisms to monitor access to, and modification of, database objects and resources. These auditing mechanisms are employed to prepare a report to list all user operations underway within database. Upon the breach of the database security prevention and detection algorithms by malicious users, the auditing techniques are employed to report all transactions. Based on the auditing report, the database administrator sends a broadcasting request. A secret sharing mechanism is applied to monitor database administrator's transactions in a lowest possible time. If the number of transactions reached n, a max necrotic signal is raised and the user is disconnected from the security system.

The main objective of dividing the signals to apoptotic and necrotic signals is to reduce the false positive (FP) alarms to the minimum. The normal user can pass a valid username and password and may enter a wrong authentic factor set and the detector fails in detecting the wrong factor. The normal user can pass an authentic user certificate authorization (UCA). As a result, he can enter the security system. If the signals are not divided, the error alarm will disconnect the user although the user is normal one. As a result, the false positive (FP) alarms increases.

C. The System Hibernation

Once user transactions are suspended, the security system must be hibernated until the suspended transactions are approved or disapproved. As presented in [18], an alternate schema is developed to obtain all transactions from normal users until the suspension process is finished. As presented in Fig. 2, when the security system verifies a suspicious user, the suspicious user transactions are suspended in the original database which resides in the database server. The security system must verify the suspicious user transactions whether to be saved in the original database or not. The time spent in the verification process will delay other transactions that are executed from normal users. This will increase the time complexity.

In order to keep an efficient, flexible, and solid security system, data hibernation is executed by transferring data from their original source to their alternate designed source. As protective measures for data hibernation, an alternate data source in compliance with the original data source should be designed. The data can be only transferred from the original source to the alternate one only if both sources are in compliance with each other [18].

The original database schema is deactivated until the super administrator (SA) verifies the suspended transactions. Normal users are diverted to the alternate database schema that is activated to allow normal users performing their transactions with the ability to keep data integrity intact.

After the super administrator (SA) verifies the suspicious user transactions, the original database schema is activated again while the alternate database schema is deactivated. Normal users' transactions are merged in the original database schema to keep the database in a consistent state.

The submitted transactions of the apoptotic and necrotic actions are suspended until the super administrator (SA) verifies the transactions. The transactions are verified to be saved in the database server or rollback. The database server sends a request to the super administrator (SA) to grant or deny the suspended transactions. In order prevent intruders from sneaking to the security system; the connection between the database server and the super administrator (SA) must be secured.

IV. Database Auditing Mechanisms and Techniques

As we presented in previous section, the database administrator performs auditing mechanisms to monitor access to, and modification of, database objects and resources. Several popular mechanisms can be deployed to audit the database structures whether the transactions are carried out by malicious users or not. If regular users execute different database transactions, a report that lists all user operations inside database is created by the auditing mechanisms. In contrast, upon the breach of the database security prevention and detection algorithms by malicious users, the auditing techniques are employed to report all transactions.

We implemented the auditing mechanisms required in most environments, namely auditing the login and logout operations inside database, auditing database operations outside normal hours, auditing data dictionary language (DDL) activities, auditing database errors that may encounter with the database security system, auditing changes to the source database if malicious users succeed in breaching the system, and auditing changes to sensitive attributes to prevent any data disclosure. Based on the auditing reports, the database administrators (DBAs) and the super administrator (SA) can execute error containment operation by restoring all malicious transactions in the database.

A. Auditing Logon/Logout into Databases

The first category for auditing database is to provide a full audit trail of any user who has signed into the database. Two events must be recorded for the auditing operation: the sign-on event and sign-off event. The following schema presents a user login history table that records all login and logout operations inside database

```
Create table user_login_audit (
user_id          varchar2(30),
session_id       number(10),
host             varchar2(30),
ip_address       varchar2(30),
login_time       timestamp,
logout_time      timestamp );
```

As presented in the previous schema, the login name for signing on as well as the timestamp for the event must be recorded. The recording process must be also applied to the TCP/IP address of the client and the program initiating the connection. Logon and logoff activities can be audited with the help of database features or the external database security solutions.

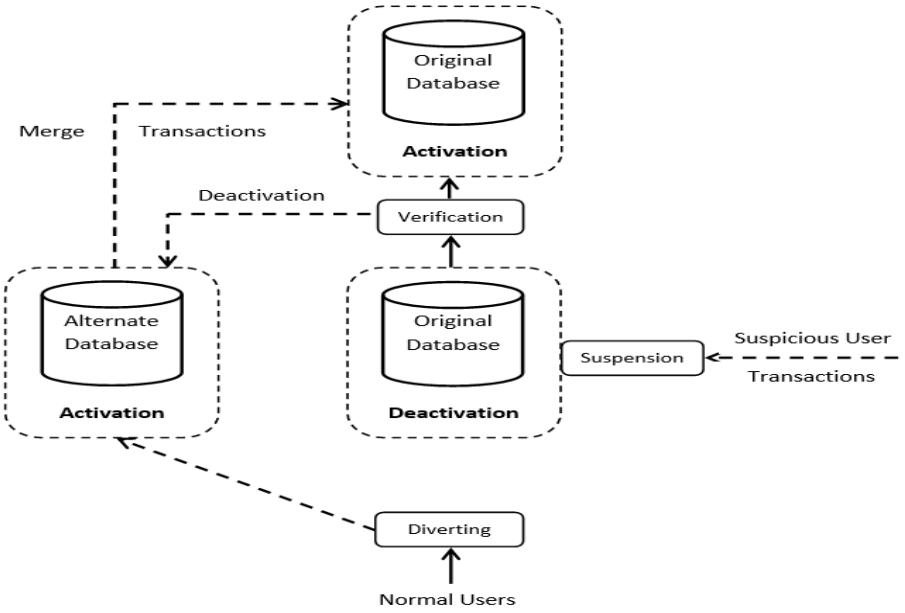

Fig. 2. System Hibernation.

In order to reduce the security cost complexity, all auditing operations are recorded using implemented database security features. In order to record all logging activities, we developed two database triggers to fire when any login attempt is initiated. The first trigger is fired at any login attempt while the second trigger is fired at any logout attempt. Table I records all login and logout attempts inside database.

```
Create or replace trigger user_login_trigger
After Logon on Database
Begin
Insert into user_login_audit
Values (User, sys_context ('USERENV',
'SESSIONID'), sys_context ('USERENV', 'HOST'),
ora_client_ip_address, Localtimestamp, null);
Commit;
End;
Create or replace trigger user_logout_trigger

Before Logoff on Database
Begin
Update user_login_audit
Set logout_time = Localtimestamp
Where sys_context('USERENV', 'SESSIONID') =
session_id;
Commit;
End;
```

As presented in Table I, all login and logout operations are recorded in the "user_login_audit" table. The super administrator (SA) needs to re-optimize all recorded information to obtain information about specific users. The super administrator (SA) can build an "audit_log_summary" table to view all usernames and the total number of logging times for each username as presented in the following schemas. The records resulted from the schemas are presented in Table II.

```
Create table user_log_summary (
User_id             varchar2(30),
Login_no            number);
Declare
Cursor C is
Select user_id, count(user_id)
From user_login_audit
GROUP BY user_id;
Begin
Open C;
Loop
   Fetch C into x, y;
   Insert into autdi_log_summary
   Values(x, y);
   dbms_output.put_line(' -User-' ‖x‖
'Connected ' ‖y‖ 'Times');
   Exit When C%notfound;
End loop;
End;


Select ' -User-'‖ user_id ‖'Connected ' ‖
sum(login_no) ‖ 'Times ' "Connection History"
From audit_log_summary
GROUP BY user_id
```

When an external security system is used, SQL firewall is used to block any connection after a given number of failed attempts by the same login name is reached. Under these circumstances, the database will not lose the connection attempts due to the rejection of the attempts at the firewall level. Instead of using external security system, database triggers are used to generate an alert following a fixed number of failed attempts as presented in Fig. 3. The alert is sent as a notification to the database administrator (DBA) to block the account. Blocking the user account requires the database administrator to join a secret sharing operation with other database administrators (DBAs) to grant or deny the operation.

All failed login attempts are recorded in the "Antigen Table Response" as presented in our intrusion detection algorithm [2]. If the same intruder attacks the system again, the security system will check the antigen table response first to detect the intruder and provides a quick response.

B. Auditing Databases outside Normal Operating

We implemented a second auditing mechanism recording activities that may be conducted beyond the regular operating business hours. From a business and a compliance perspective, this is a fundamental requirement. Auditing database usage beyond the regular business hours is critical given that off-hour activities enable unauthorized users to access or modify targeted data without suspicion. The following schemas are used to record all users who connect to the system outside normal hours. The super administrator generates a table called "operating_hours_history" which records the username, session ID, host name, and the login time for each user connected to the system outside normal hours. The records are presented in Table III.

TABLE. I. LOGIN/LOGOUT AUDITING MECHANISM

USER_ID	SESSION_ID	HOST	IP_ADDRESS	LOGIN_TIME	LOGOUT_TIME
SCOTT	20028	WORKGROUP\DESKTOP-247D553		01-SEP-19 01.50.31.226000 PM	01-SEP-19 01.50.47.468000 PM
SYSMAN	0	DESKTOP-247D553		01-SEP-19 01.50.43.245000 AM	01-SEP-19 01.50.43.256000 AM
HR	20029	WORKGROUP\DESKTOP-247D553		01-SEP-19 01.50.47.561000 PM	01-SEP-19 01.51.47.523000 PM
SYSTEM	20030	WORKGROUP\DESKTOP-247D553		01-SEP-19 01.51.01.643000 PM	
DBSNMP	20031	WORKGROUP\DESKTOP-247D553	10.66.32.16	01-SEP-19 01.51.05.643000 PM	01-SEP-19 01.51.05.943000 PM
DBSNMP	20032	WORKGROUP\DESKTOP-247D553	10.66.32.16	01-SEP-19 01.51.07.345000 PM	01-SEP-19 01.51.07.868000 PM
DBSNMP	20033	WORKGROUP\DESKTOP-247D553	10.66.32.16	01-SEP-19 01.51.09.008000 PM	01-SEP-19 01.51.09.086000 PM

TABLE. II. CONNECTION HISTORY

Connection History
User – HR – Connected – 1 Times
User – SCOTT – Connected – 1 Times
User – SYSTEM – Connected – 1 Times
User – SYSMAN – Connected – 1 Times
User – DBNMP – Connected – 1 Times

TABLE. III. SUSPICIOUS USERS OUTSIDE NORMAL HOURS

USER_ID	SESSION_ID	HOST	LOGIN_TIME
SYSTEM	30629	WORKGROUP\DESKTOP-247D553	07-SEP-19 02.41.55.481000 PM
SCOTT	30631	DESKTOP-247D553	07-SEP-19 02.44.36.057000 PM
HR	30633	WORKGROUP\DESKTOP-247D553	07-SEP-19 02.44.45.558000 PM
SYSTEM	30634	WORKGROUP\DESKTOP-247D553	07-SEP-19 02.44.54.435000 PM
SCOTT	30635	WORKGROUP\DESKTOP-247D553	07-SEP-19 02.45.01.799000 PM
SYSTEM	30636	WORKGROUP\DESKTOP-247D553	07-SEP-19 02.45.09.480000 PM

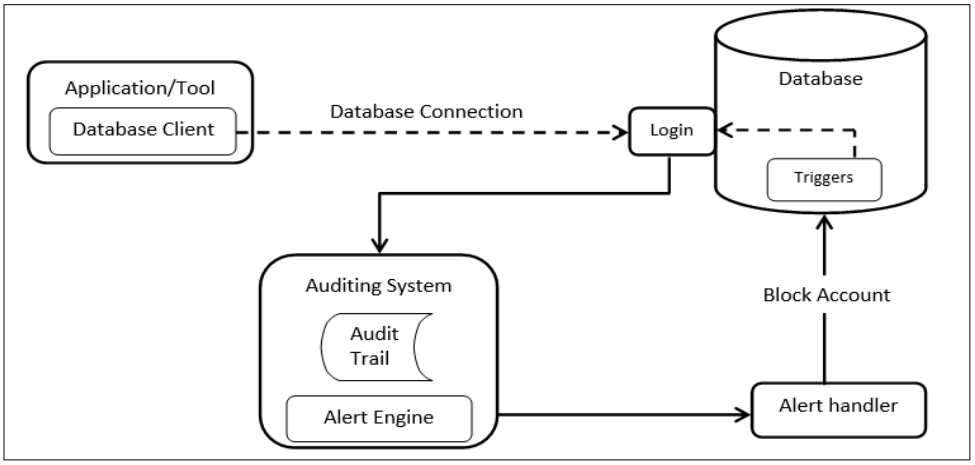

Fig. 3. Account Locking Database Procedure.

```
Create table operating_hours_summary(
user_id       varchar2(30),
session_id    number,
host          varchar2(30),
log_time      timestamp );


Create or replace trigger check_normal_hours
After Logon on Database
Begin
if  to_number(to_char(sysdate,  'HH24'))  not
between 8 and 14 OR to_char(sysdate, 'DY') in
('FRI', 'SAT')
then
Insert into operating_hours_summary
Values    (USER,    sys_context    ('USERENV',
'SESSIONID'), sys_context ('USERENV', 'HOST'),
localtimestamp);
End If;
End;
```

C. Auditing DDL Activities

Auditing data definition language (DDL) activity is considered one of the most important audit trail methodologies. The DDL commands are the most destructive as they can be exploited by intruders in order to attack the system and disclose confidential information. Many regulations require an auditing mechanism to prevent intruders from modifying the data structure such as tables or views. Three main methods for auditing schema changes exist. First: by using database audit features. Second: by using external auditing system. Third: by comparing schema snapshots. This latter method will be presented in the next subsection.

For auditing the schema changes using database audit features, the super administrator creates an audit DDL table called "audit_DDL" as explained in the following schema.

```
Create table audit_DDL(
user_id              varchar2(30),
ddl_date             timestamp,
event_type           varchar2(25),
object_type          varchar2(25),
owner                varchar2(25),
object_name          timestamp );
```

The super administrator generates a database trigger to audit all changes in schema structures and saves the changes in the "audit_DDL" table as presented in the following schema.

```
Create or replace trigger DDL_trigger
After DDL on Database
Begin
  Insert into audit_DDL
  Values (ora_login_user, localtimestamp,
             Ora_sysevent,
             Ora_dict_obj_type,
             Ora_dict_obj_owner,
             Ora_dict_obj_name);
End;
```

If the database schema changes by database administrator or users, all the changes will be recorded in the audit DDL table as presented in Table IV.

D. Auditing Changes to Database Source

As presented in the previous subsection, the first method for auditing database changes is to use database auditing features. The second method which is based on using external auditing system is costly. The third method for auditing schema changes is by using schema snapshots. The super administrator can take a snapshot from the schema source as presented in Fig. 4. By applying the Hash encryption technique (H), the source snapshot is encrypted (h1) and stored in the database server. If the malicious user modifies the source snapshot, the hash function will create a new snapshot called "suspicious snapshot" (h2). The super administrator matches the original snapshot hash (h1) with the suspicious one (h2). If there is no matching, an intrusion has happened otherwise no intrusion will be found.

E. Auditing Database Errors

Auditing errors returned by the database is among the first implemented audit trails for eliminating SQL injection, failed logins, and privilege elevation. For eliminating SQL injection, attackers may need to estimate the right number of columns. Obtaining the right number will be unlikely because the database will automatically return an error code claiming that the selected columns by the two SELECT statements do not correspond.

Another instance of an error that requires logging and monitoring is failed logins, even in the event that there are no auditing logins to the database. A failed endeavor to elevate privileges is an essential indication that an attack is underway. In order to record all database errors, the super administrator builds an audit error table as explained in the following schema.

```
Create table audit_error(
user_id      varchar2(30),
session_id   number,
host         varchar2 (30),
error_date   timestamp,
error_no     varchar2 (100),
error_txt    varchar2 (300));
```

The super administrator generates a system trigger to record all database errors in the audit error table as presented in the following schema. All recorded database errors are explained in Table V.

```
Create or replace trigger audit_error_trigger
After Servererror on Database
Begin
Insert into audit_error
Values (User, sys_context('USERENV',
'SESSIONID'), sys_context('USERENV', 'HOST'),
localtimestamp,
dbms_standard.server_error(1),
dbms_standard.server_error_msg(1));
Commit;
End;
```

TABLE. IV. DDL AUDITING TABLE

USER_ID	DDL_DATE	EVENT_TYPE	OBJECT_TYPE	OWNER	OBJECT_NAME
SCOTT	07-SEP-19 02.57.55.898000 PM	CREATE	TABLE	SCOTT	SALARY
HR	07-SEP-19 02.59.00.790000 PM	CREATE	VIEW	HR	V1
SYSTEM	07-SEP-19 03.01.01.925000 PM	ALTER	TABLE	SYSTEM	DEPT

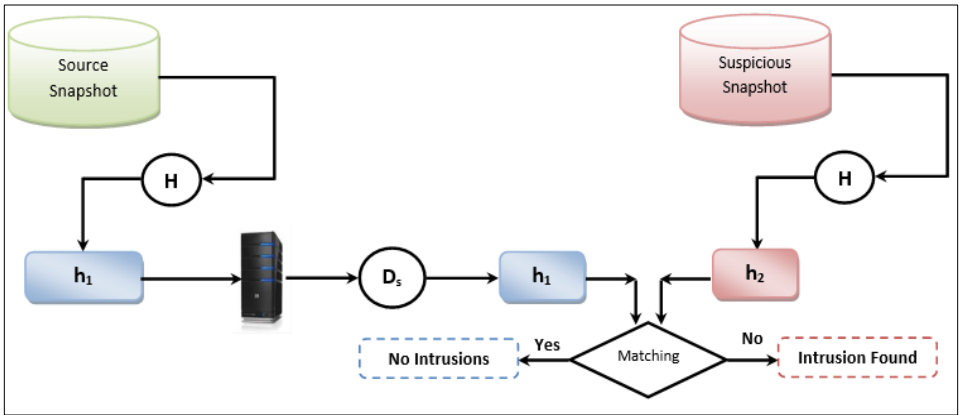

Fig. 4. Schema Snapshot Matching.

TABLE. V. ERROR AUDITING TABLE

USER_ID	SESSION_ID	HOST	ERROR_DATE	ERROR_NO	ERROR_TXT
SCOTT	30646	WORKGROUP\DESKTOP-247D553	07-SEP-19 03.03.08.490000 PM	942	ORA-00942: table or view does not exist
SCOTT	30646	WORKGROUP\DESKTOP-247D553	07-SEP-19 03.03.52.917000 PM	1	ORA-00001: unique constraint (SCOTT.PK_EMP) violated
HR	30648	WORKGROUP\DESKTOP-247D553	07-SEP-19 03.04.53.040000 PM	942	ORA-00942: table or view does not exist
HR	30648	WORKGROUP\DESKTOP-247D553	07-SEP-19 03.05.19.066000 PM	1031	ORA-01031: insufficient privileges

F. Auditing Security Attributes Changes

Auditing DML activity is an important requirement in database auditing systems. The auditing operation requires recording the old and new values for each DML activity. Two different requirements must be implemented to fully record the DML activity. First: to record any update operation, the user who has performed the update, which session has been used, and the time for the DML activity. Second: to record what the value was before and after the update operation.

The first requirement for recording the DML information is developed by building a DML audit table as presented in the following schema.

```
Create table DML_audit(
username          varchar2(20),
session_id        number,
host_name         varchar2(40),
insert_time       timestamp,
update_time       timestamp,
delete_time       timestamp
);
```

The super administrator develops a DML trigger that records all DML operations and stores the result in the DML audit table as presented in the Table VI.

```
Create or replace trigger DML_trigger
After Insert or Update or Delete On Scott.emp
For each row
Begin

If Inserting Then
Insert into DML_audit
Values (User, sys_context('USERENV',
'SESSIONID'), sys_context('USERENV', 'HOST'),
localtimestamp, Null, Null);

Elsif Updating Then
Insert into DML_audit
Values (User, sys_context('USERENV',
'SESSIONID'), sys_context('USERENV', 'HOST'),
Null, localtimestamp, Null);

Elsif Deleting Then
Insert into DML_audit
Values (User, sys_context('USERENV',
'SESSIONID'), sys_context('USERENV', 'HOST'),
Null, Null, localtimestamp);

End if;
End;
```

The second requirement for recording the DML information is developed by building an audit change table to record the updated value before and after the update operation as presented in the following schema.

```
Create table audit_changes
(
username          varchar2(20),
DML_time          timestamp,
oldempno          integer,
newempno          integer,
oldname           varchar2(20),
newname           varchar2(20),
oldhiredate       date,
newhiredate       date,
oldsal            number,
newsal            number,
oldcomm           number,
newcomm           number
);
```

The super administrator develops a DML trigger that records all old and new DML values and stores the result in the audit change table as presented in the Table VII.

```
Create or replace trigger
trigger_table_changes
After Insert or Update or Delete On Scott.emp
For each row
Begin
Insert into audit_changes
Values (User, localtimestamp, :old.empno,
:new.empno, :old.ename, :name.ename,
:old.hiredate, :new.hiredate, :old.sal,
:new.sal, :old.comm, :new.comm);
End;
```

Suppose that 1 million DML transactions are executed per day and each transaction updates a single value. The database contains 100 tables and each table contains 10 attributes. The database contains 10,000 records in each table. If the super administrator (SA) develops an auditing system that records all attributes changes before and after the update activity, then the database will grow 35 times larger than the original database after one year.

As presented in [19, 20, and 21], the DML activities must be recorded for sensitive attributes only in order to reduce the space complexity of database size.

TABLE. VI. DML ACTIVITY TIME

USER_ID	HOST_NAME	INSERT_TIME	UPDATE_TIME	DELETE_TIME
SCOTT	WORKGROUP\DESKTOP-247D553	07-SEP-19 03.08.30.853000 PM		
SCOTT	WORKGROUP\DESKTOP-247D553			07-SEP-19 03.09.51.031000 PM
SCOTT	WORKGROUP\DESKTOP-247D553		07-SEP-19 03.08.56.949000 PM	

TABLE. VII. DML AUDITING VALUES

USERNAME	DML_TIME	OLDEMP-NO	NEWEM-PNO	OLDENA-ME	NEWENA-ME	OLDHIRE-DATE	NEWHIR-EDATE	OLDSAL	NEWSAL	OLDCOM-M	NEWCO-MM
SCOTT	07-SEP-19 03.14.12.399000 PM	7369	7369	SMITH	SMITH	17-DEC-80	17-DEC-80	800	800		
SCOTT	07-SEP-19 03.14.12.403000 PM	7566	7566	JONES	JONES	02-APR-81	02-APR-81	2975	2975		
SCOTT	07-SEP-19 03.14.12.403000 PM	7788	7788	SCOTT	SCOTT	19-APR-87	19-APR-87	3000	3000		
SCOTT	07-SEP-19 03.14.12.403000 PM	7876	7876	ADAMS	ADAMS	23-MAY-87	23-MAY-87	1100	1100		
SCOTT	07-SEP-19 03.14.12.403000 PM	7902	7902	FORD	FORD	03-DEC-81	03-DEC-81	3000	3000		
SCOTT	07-SEP-19 03.14.37.229000 PM	7654	7654	MARTIN	MARTIN	28-SEP-81	28-SEP-81	1375	1650	1400	1400

G. Auditing Changes to Privileges, Users and Roles

The final auditing category is to keep a complete audit trail of any changes to the privileges, users, and roles. Different categories must be recorded to monitor the user activities in database. First: addition and deletions of users and roles. Second: privilege changes. Third: change to the security attributes at a server, database, statement, or object level.

As presented in [19, 20, and 21], the first and second categories are secured by using the secret sharing algorithm. A single database administrator (DBA) cannot add or delete users and roles, or modifies user privileges without the agreement of other database administrators (DBAs) according to the super administrator (SA) infrastructure.

The third category which is based on protecting security attributes from modification is based on preventing a single database administrator (DBA) from changing the sensitive and most sensitive attributes as presented in [2].

V. CONCLUSION

Artificial immune system (AIS) is a cover term for all the attempts that develop computational models in the spirit of biological immune systems. This paper presents an error containment algorithm as a post-security countermeasure for detecting malicious intrusions. A system hibernation framework is embedded with the proposed algorithm to monitor users' transactions. Different auditing mechanisms are implemented to track the users' behaviors whether they are authorized or not. Based on the results of the auditing mechanisms and users' authorizations, the transactions are committed or rolled back.

As short-term future work, we plan to implement the proposed artificial immunity-based algorithm and evaluate its accuracy by comparing our experimental results to those of post-security algorithms identified in the literature. The accuracy of the algorithms will be evaluated based on reducing the false positive and false negative alarms.

Over a medium-term research perspective, we propose to apply the artificial immunity-based algorithm on cloud service providers using different cloud deployment models.

REFERENCES

[1] C.M. Ou, “Host-based intrusion detection systems adapted from agent-based artificial immune systems”, International Journal of Neurocomputing, ELSEVIER, vol.88, 2012, pp.78-86.

[2] A. M. Mostafa, N. Yanes, and S. A. Alanazi, "A Cognitive adaptive artificial immunity algorithm for database intrusion detection systems", Journal of Theoretical and Applied Information Technology, vol. 97, no. 16, 2019, pp. 4387-4400.

[3] D. A. Fernandes, M. M. Freire, P. A. Fazendeiro, and P. R. Inácio, “Applications of artificial immune systems to computer security: a survey”, ELSEVIER Journal of Information Security and Applications, vol. 35, 2017, pp. 138-159.

[4] D. Dipankar, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications", Applied Soft Computing, vol. 11, no 2, 2011, pp. 1574-1587.

[5] S. Ariffin, R. Mahmod, A. Jaffar, and M. R. K. Ariffin, “Immune systems approaches for cryptographic algorithm”, IEEE International Conference on Bio-Inspired Computing: Theories and Applications, 2011, pp.231-235.

[6] F. Hosseinpour, K. Abu-Baker, A. H. Hardoroudi, and N. Kazazi, “Survey on artificial immune systems as a bio-inspired technique for anomaly based intrusion detection systems”, IEEE International Conference on Intelligent Networking and Collaborative Systems, 2010, pp.323-324.

[7] Z. Cui, X. Lu, and J. Wang, "Adaptive intrusion tolerance strategy of the system based on artificial immune", IEEE International Conference on Computational Intelligence and Software Engineering, 2009, pp.1-4.

[8] M. Puteh, K. Omar, A. R. Hamdan, and A. Abu-Bakr, "Classifying heterogeneous data with artificial immune system", IEEE International Symposium on Information Technology, 2008, pp.1-5.

[9] L. Wang, C. Yin, and H. Dong, "A Novel generalized framework for access control based on the immune mechanism", IEEE International Conference on Intelligent Control and Automation, 2008, pp.1427-1431.

[10] D. Dal, S. Abraham, A. Abraham, S. Sanyal, and M. Sanglikar, "Evolution induced secondary immunity: an artificial immune system based intrusion detection system", IEEE International Conference on Computer Information Systems and Industrial Management Applications, 2008, pp.65-70.

[11] K. Chen, G. Chen, and J. Dong, "An Immunity-based intrusion detection solution for database systems", International Conference on Web-Age Information Management, Springer, Berlin, Heidelberg, 2005, pp.773-778.

[12] H. Wang, "A Security framework for database auditing system", IEEE International Symposium on Computational Intelligence and Design, vol. 1, 2017, pp.350-353.

[13] K. Wu, L. Hua, X. Wang, and X. Ding, "The Design and implementation of database audit system framework", IEEE International Conference on Software Engineering and Service Science, 2014, pp. 553-556.

[14] A. A. Elshiekh, and P. D D. Dominic, "Three audit stages for securing statistical databases", IEEE International Conference on Information Management and Engineering, 2009, pp.283-286.

[15] Shao Zuozhi, Li Yunpeng, Zhang Kuo, Zeng Geng, Zhao Sitang, "An Audit Method Based on Mathematical Statistics Detection in Database Audit System", IEEE Conference on Intelligent Human-Machine Systems and Cybernetics, 2015, pp. 203-206.

[16] Q. Huang, and L. Liu, "A Logging scheme for database audit", IEEE International Workshop on Computer Science and Engineering, vol. 2, 2009, pp.390-393.

[17] L. Wentian, G. Miklau. "Auditing a database under retention restrictions", IEEE International Conference on Data Engineering, 2009, pp.42-53.

[18] S. Safdar, M. F. Hassan, M. A. Qureshi, and R. Akba, "Data hibernation framework in workflows under intrusion threat", IEEE Symposium on Computers and Informatics, 2011, pp.680-685.

[19] A. M. Hashem, I. M. El-Henawy, and A. M. Mostafa, "Interactive multi-layer policies for securing relational databases", IEEE International Conference on Information Society, 2012, pp.65-70.

[20] A. M. Mostafa, A. M. Hashem, and I. M. El-Henawy, "Design and implementation of multi-layer policies for database security", International Journal of Information Sciences, Natural Sciences, vol. 2, no.3, 2013, pp.147-153.

[21] A. M. Mostafa, A. M. Hashem, and I. M. El-Henawy, "Design and implementation of extensible service-oriented algorithms for securing relational databases", International Journal of Digital Content Technology and its Applications (JDCTA), Elsevier, vol.7, 2013, pp.753-763.