A novel machine learning approach for database exploitation detection and privilege control
Chee Keong Wee and Richi Nayak

ABSTRACT
Despite protected by firewalls and network security systems, databases are vulnerable to attacks especially when the perpetrators are from within the organization and have authorized access to these systems. Detecting their malicious activities is difficult as each database has its own set of unique usage activities and the generic exploitation avoidance rules are usually not applicable. This paper proposes a novel method to improve the security of a database by using machine learning to learn the user behaviour unique to a database environment and apply that learning to detect anomalous user activities through the analysis of sequences of user session data. Once these suspicious users are detected, their privileges are systematically suppressed. The empirical analysis shows that the proposed approach can intuitively adapt to any database that supports a wide variety of clients and enforce stringent control customized to the specific IT systems.

1. Introduction
The demand for database services in the corporate world is growing rapidly. This creates a challenge for IT administrator to manage databases with limited labor and resources. One of the challenges is to oversee the security for these databases, especially when an organization needs to manage hundreds or thousands of databases (Connolly & Begg, 2006). Contemporary databases come with their own inbuilt security feature (Bertino & Sandhu, 2005). However, there are several challenges that IT administrators must handle such as volume and variety of databases as well as the individual database complexity (Hoffer, Ramesh, & Topi, 2015). Manual policy checks across such a large landscape of database services are both difficult and time-consuming (Connolly & Begg, 2005). There are commercial IT productivity software tools that can help to watch a database system and improve the security landscape even further. However, challenges such as exploitation by internal staffs are much harder to detect (Khanuja & Adane, 2011). Existing software use audit features and rules-based monitoring to oversee the activities in the database. Unfortunately, they have been developed to handle a wide range of database’s vulnerability and may not be able to keep up with the complex intrusion and exploitation methods (Khanuja & Adane, 2011), (Sturrus & Kulikova, 2016).
With the presence of user logins in databases, it is open to vulnerability attack and raises a security concern (Bertino & Sandhu, 2005). While the databases’ internal security system addresses to an extent, some organizations attempt to be more secure by granting no privileges to users in the databases or by removing user accounts (Liu et al., 2004). Given that most IT departments manage a big range of heterogeneous systems, the challenge will be even greater (Bertino & Sandhu, 2005). There are commercial software solutions like Identity Access Management (IAM) that mostly work on a centralized user authentication and privileges or roles associations (Kruk et al., 2006), (Gaedke, Meinecke, & Nussbaumer, 2005) where users are firstly authenticated before they can access the databases. Users, roles, and privileges are present inside the databases as part of the IAM multi-tier architecture deployment (Bertino & Sandhu, 2005). This strategy works well especially when the main authentication and access are managed from a single server. However, hackers are becoming more sophisticated and have their ways to bypass the conventional authentication method (Liu & Chen, 2003). In order to remain safe, some organizations give the least privileges to the users and make it harder for hackers (Ramaswamy & Sandhu, 1998).

All the existing database security monitoring systems only perform post-activity reporting and monitoring, while the IT administrator covers the responsibility of overseeing the user account control and rectify problems whenever they are discovered. This challenge gets harder in proportion to the number of databases that they must oversee (Bertino & Sandhu, 2005). The databases do have their own security features for users and privileges against the schemas and objects, whereas central authentication systems can consolidate the security into a single point of control such as Lightweight Directory Access Protocol (LDAP) or Microsoft’s Active Directory are used (Lampson, 2004). Even when these security features are deployed, there is a tiny chance that they can be compromised as there are unconventional techniques such as security database exploits that have been documented to hack through or bypass them (Pritchett & De Smet, 2013), (Gaetjen, 2015). Databases such as PostgreSQL, Oracle, MySQL or Firebird have their own internal security systems which are difficult to manage individually by the IT administrators in the absence of central authentication or access control mechanisms (Bertino & Sandhu, 2005).

Machine learning such as deep neural network has gained a lot of attention in the application to cybersecurity in recent times as security breaches and threats are getting sophisticated over times and a highly responsive machine learning approach is needed to counteract against them. Authors in (Trejo, Clempner, & Poznyak, 2018) proposed the use of reinforcement learning (RL) for Stackelberg security games through deploying attackers versus defenders strategies against the environment in order to win and they present the application of their proposed RL approach to several security scenarios. Authors in (Buczak & Guven, 2016) use RL to monitor and analyze the autonomous vehicle’s dynamics as it is subjected to cyber and physical attacks on its sensory devices and systems. Then use RL to learn and find the best approach to rebel against the cyber-attacks. Its uses a combination of deep q-learning and long-short term neural network to find the maximum possible deviation and seek the optimum solution. Another in (Partha, Bardhan, Chowdhury, & Phadikar, 2017) use a similar approach where the RL has been the foundation for the author’s approach to detect and counteract against cyber-attacks in the cloud computing domain. Based on these finding, it is proper that RL should be considered to enhance database security.
This paper proposes a novel approach of detecting the database exploitation as an anomaly by analyzing user activities in the database with machine learning methods and then act to nullify the threat. We propose to enhance the current databases’ security management by performing the role of a central access control system as well as removing the presence of privileges against the database’s objects. This takes the security measure a step further up in the security landscape. As each IT systems have their own distinctive operations, activities, and usage, conventional detecting systems cover common anomalous detecting rules and require customization for them to fit the target DB environment. Machine learning techniques are used to learn the behaviour of the users within the target DB to gain insight about their activities, then exploit that experience to determine subsequent new connecting DB users if their action is not normal. We conducted experiments against a non-production IT system where it starts by learning against the database in a controlled state which have been captured for a week period. Then we introduce several outlier activities into the database to assess if the system can detect them. Empirical analysis reveals that the proposed approach can identify user activity anomalies and take proactive action against them by revoking their privileges.

2. The proposed approach

We propose a novel approach that can detect anomalous activities in a database and act to remove the threats after they have been identified. The proposed approach has two modules; Database Exploitation Detection (DED) and Privilege Control System (PCS). Figure 1 shows a high-level view of the users’ interaction with the proposed modules and the target database. Before the users logged into the database, they will be authenticated by the database’s security and PCS which takes care of the privilege control in the database. DED is the intelligent module that performs the task of user activities monitoring. When DED detects anomalies, it will notify the IT administrator about the incident and initiate PCS to terminate the anomalous user session immediately. DED performs anomaly detection based on the analysis of the entire sequence of data observed using the following information such as the machine that the user used to log in, the type of application that they used, the login time in terms of days and hours, database objects queried and

![Figure 1. Proposed DED and PCS architecture.](image-url)
the SQL that they executed. PCS performs user authentication through a set of pre-determined policies that have been set by the IT administrators and proactively practices self-healing approach. Self-healing is a security enhancement method where the accounts are created and exist only for the period when the users need to use the database. They are removed once they log out and their DB privileges are dropped (Liu et al., 2004). The idea is to minimize the number of security prospects that a potential hacker may want to exploit. Typically, IT administrators pre-create user accounts across all the databases and leave them there for convenience. But that exposes the database unnecessarily. If a user sets his password to a simple phrase, that is easy to guess, it poses a security risk.

### 2.1. Database exploitation detection (DED) module

DED uses an ensemble method of association rule mining and reinforcement learning to monitor the user activities continuously and to identify the anomalous activities in the database. The DED module starts on the premise that it has little or no prior knowledge available about the database or how the users use it. The knowledgebase is collected by observing the activities and learning what activities constitute as an anomaly. It applies this knowledge while monitoring the database and samples new user sessions continuously.

A user session has many types of data fields. In this research, the following attributes (as shown in Table 1) were chosen to analyze anomalous activities. They are; login time, machine name, OS user account, Terminal, the application used, and SQL ID. SQL ID is a hash value in which it refers to the SQL statement or code that the user is executing. Based on this information, the DED module can mine knowledge on the exact schema, objects, and dynamic manipulating languages (DML) that the users are executing. The data are collected over a period covering both working days and weekends to capture a sampling size that can be sufficiently representative of the work activities.

Figure 2 illustrates the process of acquiring the data and extracting the useful information to build up the knowledge-base.

### 2.2. Association rule mining

The challenge in detecting anomalies among user activities in the database is the fact that it is an open problem without clearly defined limits or boundary. The complexity also lies in the modelling of the sequence of related data that have varying temporal patterns that

| Logon_Time       | Machine     | OSUser  | Application | Terminal | SQLID          |
|------------------|-------------|---------|-------------|----------|----------------|
| 29/03/2017 17:25 | Q225491     | ja055   | toad.exe    | Q225491  | 53c2dfrtgu595  |
| 30/03/2017 12:46 | CBNEWAS14   | SVC-SAPIS | process.exe | CBNEWAS14 | -              |
| 30/03/2017 13:21 | SPRQQ229065 | cv073   | sqldeveloper | Q229065  | 8dhxuxdbrngju  |
| 29/03/2017 4:37  | cbns1db02   | qeedwhpr | Datastg     | cbns1db02 | 4fhkwpwv28qt   |
| 29/03/2017 14:46 | sbnfwu15    | svc_SAP..T | websvc     | sbnfwu15  | -              |
| 30/03/2017 12:24 | cbns1db02   | qeedwhpr | sqlplus     | cbns1db02 | 8dhxuxdbrngju  |
| 29/03/2017 17:25 | Q225491     | ja055   | sqldeveloper | unknown  | 4q0js1p8mv2wp  |
| 30/03/2017 4:13  | cbns1db02   | qeedwhpr | websvc     | cbns1db02 | 4q0js1p8mv2wp  |
| 30/03/2017 7:43  | cbns1db02   | qeedwhpr | websvc     | cbns1db02 | 53c2dfrtgu595  |
make it difficult to predict. We propose to apply association rule mining (Tajbakhsh, Rahmati, & Mirzaei, 2009) to learn the relationship in the user sessions data and build up knowledgebase to differentiate the pattern of commonly acceptable user behaviour for a specific system. There is a strong correlation between users’ activities and their intention to use. Their behaviours are reflected in the relationship learned from the data such as applications used, machine name, schema queried, DML executed, login time and objected accessed. Figure 2 shows a basic overview of the process flow of using the association rules to define and predict the DB hackers. Figure 3.

To use association rules for our proposed system, the method mines through the data batch of user activities which have been captured from the production databases over a period of several business cycles. The data batch size must be comprehensive enough to have sufficient representation of all the possible normal user activities with some potential anomalous one included. The system will build up a repository of all the association rules across the user data records which form the baseline of the next step. We chose the Apriori
algorithm (Rahman, Ezeife, & Aggarwal, 2010) to generate associate rules. Let the data items be labelled as $A$, $B$, etc. A set of these items is called an itemset which we called it $X$. An association rule between the item sets is expressed as, $A \rightarrow B (c, s)$, where $A$ and $B$ are items from the DB session and they are a subset of the item-sets $X$; $(A, B \subset X)$, $c$ and $s$ are confidence and support of the rule. Support indicates how frequent the itemset appear in the data set and is expressed as a percentage of the items in the session repository, $D$, which contain Item-sets $A$ and $B$ in equation (1). Confidence is expressed as a percentage of support of Item $A, B$ over the support of $A$ in equation (2). Lift is the percentage of the support of item $A$ and $B$ if $A$ and $B$ are independent.

$$\text{Support}(A \Rightarrow B) = P(A \cup B) \quad (1)$$

$$\text{Confidence}(A \Rightarrow B) = P(B|A) = \frac{P(A \cup B)}{P(A)} \quad (2)$$

$$\text{Lift}(A \Rightarrow B) = \frac{P(B|A)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)} \quad (3)$$

The min support threshold set the difference between frequent and non-frequent itemset where the min confidence constraint is used to those frequent itemset to form rules. The lift value determines the strength of the occurrence of the user’s data. We propose to use the lift value to rank the association rules. The higher ranked rules are considered as normal user sessions. The premise is that frequent occurrence of common normal users will have a much higher lift as compared to the abnormal ones. These user sessions provide the labelled data used in the next step as shown in Table 2. SchemaName is the owner of the database objects that the user is accessing. DBUser is the account in the databases that the user used to log in. The hostname is the name of the

**Figure 3.** Proposed Privilege Control System.
machine that the user is accessing from. The program is the name of the application that
the user used to access the database. Timing refers to the time that the user is performing
his tasks. This will be converted to symbolic representation or discretized into weekday +
hours by pre-processing.

Note that there are very few variations among the data fields that could set one ses-
sions classification to be different from another. We consider the top schema level of
DB user records for the association rules pattern mining. Some of the login information
forms the essential feature of a session while others are an extension of the information
that forms the correlation of the feature like schema access, DML performed, resource
groups, time of access. This forms the action feature of the session. The user activities
data are regarded as item-sets that comprised of items from the user login session. An
example of the association rules created on the login data is detailed in Table 3 using
data from the database’s logon as in Table 2.

### 2.3. Reinforcement learning: basic

The reinforcement learning model comprises an agent interacting with an environment as
shown in Figure 4. The agent perceives the state, s, from the environment at the time, t,
and initiate an action, a. After receiving the action, the environment responds with a
new state, st+1, plus a scalar score or reward, r (Russell, 2016). The reward can be favourable
or unfavourable depending on the result. This cycle repeats until either the goal has been
achieved or the iteration limit has been achieved. There is also a discount factor, γ, for the
agent whether to put more emphasis on currently available rewards or have more con-
sideration for the future rewards of future actions and states. The objective here is for
the RL model to get the optimal policy, Π*, with the agent able to achieve the

### Table 2. User login session and associated risk flag.

| SchemaName | DBuser | HostName | Program | Time | SQL | LIFT threshold |
|------------|--------|----------|---------|------|-----|----------------|
| SPLX2230   | qenetdpr | erokuv90 | sp_ordr@erokuv90 | 04–09 | Select … | >0.9 |
| FDRSTAT    | fcalkiosk | SOE02197 | FCLLite.Net.exe | 04–10 | Update … | >0.9 |
| EIMUPDATE  | svc-WL-EIM | ERRDAS37 | JDBC Thin Client | 04–10 | Insert … | >0.9 |
| EIMUPDATE  | svc-WL-EIM | ERRDAS37 | JDBC Thin Client | 04–09 | Exec … | >0.9 |
| SYS        | root | Unknown | JDBC Connect Client | 04–23 | Exec… | <0.1 |
| SPLX2105   | qenetdpr | erokuv90 | sp_ordr@erokuv90 | 04–12 | Select… | >0.9 |
| EDW_LDG    | SM203 | Q39484 | JDBC Thin Client | 04–13 | Select | >0.9 |
| OMSMAIN    | DP023 | ERRDAS37 | JDBC Thin Client | 04–02 | Delete … | <0.1 |

### Table 3. Example of association rules from the database login data.

| Association rule | Interpretation |
|------------------|----------------|
| Schema = EDW_LDG → DBuser = cw073 → program = sqlplus → hostname = Q223069 → timing = normal → privilege = support_role | The osuser, cw073, is logging in during normal working hour from a valid terminal with a valid program. This is considered a normal database acceptable behaviour. |
| Schema = EDW_VIS_OWNER → DBuser = root → program = putty → hostname = Q22396 → timing = offpeak → privilege = dba | The schema is on the list, the osuser is of exceptional high OS privilege, the osuser & program are within the boundary, but the timing & privilege is of concern. All the items here are of exceptionally high privilege that is not normal. |
| Schema = SYS → DBuser = oracle → program = sqlplus → hostname = unknown → timing = offpeak → privilege = sysdba | |
maximum the cumulative reward over a period. The method that we are using here is called Q-learning (Van Hasselt, Guez, & Silver, 2016) and it defines a quality function, \( Q(s,a) \), which equate to a reward to the agent for a state, \( s \), after performing the action, \( a \). The value can be derived by the Bellman’s equation which defines \( Q(s,a) = \) immediate reward + discounted future reward in equation (4) (Szepesvari, Brachman, & Dietterich, 2010).

\[
Q(s, a) \leftarrow r + \gamma \max_a Q(s', a')
\]  

(4)

Where \( s \) is the state, \( a \) is the action, \( s' \) is the future state, \( a' \) is the future action. \( r \) is the current reward, \( \gamma \) is the discount factor of the future reward that is derived from future Q function of the new state, \( s' \), and action, \( a' \). The RL model starts randomly at the beginning and as it iterates the Q-function over time, it will start to converge to an optimum Q-function called, \( Q^*(s,a) \) (Mellouk, 2011). Therefore, the entire process iterates and is driving by the optimal policy as in the equation (5);

\[
\Pi^*(s) \leftarrow \arg\max_a Q^*(s, a)
\]  

(5)

If the number of states and actions are small in numbers, then the above optimum value function and policy can be used. But in the field of database’s anomaly detection, there are many possible states and actions which cannot simply be met by the equation. The number of combinations among the variables of database sessions is astronomical, and that can be described as a curse of dimensionality problem. One way to resolve this is to use a neural network which can accept the states’ inputs and produce the possible Q-value as shown in Figure 5. To simplify our approach, we focus on current reward and equate reward to Q-value. So, the predicated Q-value from the NN versus the real Q-value will form the loss function for the NN (6);

\[
L = \frac{1}{2}[(r + \gamma \max_a Q(s', a')) - Q(s, a)]^2
\]  

(6)

In our implementation, the iteration within the NN to produce the predicted optimum Q-value will repeat until the reward or Q-value meet the requirement of \( \max_a Q(s',a') \). The predicted action is then re-validated by the agent to derive the real reward. The validated information of \( Q(s,a,r,s') \) is then added to the knowledge-base for the next round of NN training.
For our proposed system, the algorithm for Q-learning begins with the initialization of the table $Q(s, a)$ to zero for all state-action pairs $(s, a)$. It observes the state, $s$, which is the input of variables at the beginning followed by repeating the iteration of the actions as shown in algorithm 2 until it converges. First, it receives the state, $s$, from the environment and makes a prediction of the $Q(s,a)$ value using the trained NN. With the $Q(s,a)$ value, the RL agent takes the necessary action against the environment which yields a reward, $r$, and the new state, $s_{t+1}$. The next step is to add a new state, $s_{t+1}$, together with the action and reward into the knowledge-base which is again used to retrain the NN. In the subsequent iteration, the RL agent acquires the new $Q(s,a)$ value and performs the new set of training for the NN which in turn yields another set of state, action, and rewards. This keeps repeating until iteration is reached. The process of prediction, validation, and training keeps repeating until the NN can predict the $Q(s,a)$ accurately. Once the error between the prediction versus the actual value converges to the global maxima, then it is assumed that the optimal policy has been achieved (Performing Flashback Recovery, 2013).

2.4. The reinforcement learning algorithm

Referring to Figure 6, the reinforcement learning process for the DED is a progression of machine learning that can be grouped into 3 stages where the agent performs in accordance to the availability of the knowledge about the environment. Starting with the premise that the agent has no prior knowledge about the environment’s state or action, it must learn through trial-and-error through the initial learning state, the agent builds up its knowledge about the cause-and-effect between the states, actions, and rewards. The initial learning is achieved using the Apriori algorithm (Rahman et al., 2010) that has been applied to a data set which comprised of captured user session from the production database in a controlled manner. The initial action of classification is formulated by the consideration of each session against the mined association rules based on the threshold of confidence and lift values. The NN is trained to recognize the common feature and predict the lift value, which is correlated with the prediction of the commonality of the user session.

Once it builds up an initial knowledge on environment’s response to various actions, the RL agent moves to the next stage where it is considered as a semi-learned model. Here it uses the Q-learning algorithm (Van Hasselt et al., 2016) by building a model where the agent can predict the classification of a new set of user sessions based on
the knowledge generated from past train data set. The agent uses the NN model as shown in Figure 5 to predict the possible rewards and classification for the states it encounters. Normally, the NN model must undergo several repetitions on the model predictions until the reward meets or exceeds the desired threshold, thus finding the $\text{argmax}(Q(s^n, a^n))$ of rewards. In our system, the Q value determines the prediction on the state of user activity. It does require the IT administrator to authenticate those that have been classified as anomalous. The agent will then try to score the predicted actions against the environment to determine the resulting state and actual reward, then add into the training knowledge for the NN to enrich the dataset and in turn used to predict better outcomes. This process of trial-and-error is repeated until the agent has acquired sufficient knowledge about the database’s environment that it is now known to differentiate between what is considered normal user activities versus what is anomalous. Algorithm 1 is the overall RL agent’s algorithm.

3. Privilege control system (PCS)

The proposed Privilege Control System (PCS) follows the Least Privilege Principle (Schneider, 2003) where the database starts off initially with the bare essentials such as schemas, objects or features to support the application. It provides the absolute minimum privileges for the users to do their jobs. If they want more privileges, they must apply for them. After the positive assessment, the higher permissions will be granted (Tajbakhsh et al., 2009). PCS is a multi-tier architecture with agents running alongside the database and maintain a persistent connection that monitors the user session continuously. The PCS will only grant access to the users when they meet the login criteria; an hour of logon, approved
machines, approved a list of software/application to access. The knowledge-base contains
the list of users and the databases that they are approved to use, including the details such
as schemas and objects. PCS will then create the necessary individual user accounts on the
database and grant the object privileges to them, before allowing the user to initiate their
session to the database. At the same time, PCS prepares another set of commands for
revocation and they are used to remove all of the user accounts and privileges when
the users have completed their work. The second part is crucial in protecting the database
as it removes accessibility in term of user accounts to minimize the exposure to hacking.
This concept is often referred to as Self-healing database (Liu et al., 2004).

Algorithm 1: Autonomous database exploitation detection algorithm.

| Line | Description |
|------|-------------|
| 1    | Initialization 1: set value for learning, reward preference & exploration rate, the threshold for knowledge exploration, learning, and exploitation |
| 2    | Initialization 2: initialize memory, q-table collection and respective counters |
| 3    | Acquire sample batch of users’ session from the production databases |
| 4    | Loop the iteration process |
| 5    | Check the learning rate. |
| 6    | If learning <= low_learning, do the exploration phase |
| 7    | Read in a new batch of users’ sessions histories |
| 8    | Generate association rules against user session |
| 9    | Apply the rule of classifying rules’ lift value that does not meet the threshold to be anomalous. |
| 10   | Apply the action of classification |
| 11   | Find the reward of classifications |
| 12   | Add the knowledge of the state of users session, the action of classification, the reward of detection to the knowledge-base, training data |
| 13   | | |
| 14   | If learning is > med_learning and < high_learning, then do learning phase |
| 15   | Get the current state of user session from the environment. |
| 16   | Find best classification reward and action based on the current state |
| 17   | Train the NN model with the data from the knowledgebase, with the state as input and action as an output. |
| 18   | Call the NN model to predict the possible action of classification |
| 19   | Validate the action against the environment and get a new reward |
| 20   | Add the information to the memory and minibatch |
| 21   | Find the Q(s,a) for the state and action with consideration from gamma, then add them to Q-table |
| 22   | If learning > high_learning, do exploitation phase. |
| 23   | Acquire the current state from the environment |
| 24   | Get the best action from the Q-table based on q-value |
| 25   | Apply action to the environment |

Figure 3 shows the overview of the multi-tiered PCS where it maintains a persistent
session to the databases to the central control unit via software agents. One of the key
components that can help the PCS to capture all the initiation of user activities is the data-
bases’ login triggers. They capture session details of a user such as a login time, client
machine used and type of application. This information is important as they form a combi-
nation of criteria that the PCS has enforced. An example is where a user that has been
authorized to access the database but the machine, hour to login or the program that he
used has not been authorized to use, will have his session terminated (Bertino & Sandhu,
2005), (Kuhn, 2013). Creation and removal of the user account in the database is part of the
PCS process.

The next part is the privileges grant. Before a user can retrieve or manipulate data in a
table, he must have privileges such as Insert, delete, update or select against it. The PCS
grants the privileges to the list of objects that the user has been authorized to use. And once the user has completed his task and logs off, the PCS will run another set of scripts to revoke his privileges. But some user accounts are exempted from this control and these accounts usually must perform some vital tasks continuously in the background such as replication, extraction-transformation-loading, scheduled events, and jobs. Another feature in PCS is that all the resource groups are ring-fenced to minimize the possibility of privileges overlapping (Sandhu & Samarati, 1994). Figure 7 illustrates the sequences of activities that happen between databases and PCS. Note that all the procedures are handled by individual databases’ internal packages and functions; login triggers, PLSQL packages to database specific features like spatial, XML, etc.

3.1. User’s privilege representation

Typical databases have an intricate complex relationship between objects and privileges (Connolly & Begg, 2005). It is very common that certain database objects have large dependencies on other objects which require a long series of privileges that cascade down and can cross over to multitudes of other schemas boundaries or database sources. Figure 8 illustrates a typical object that has such dependencies on other objects. Such complex hierarchies are common among systems and this poses a challenge to the PCS concept; that is to have a proper representation of the hierarchical series of privileges in proper SQL format which can be created or granted as well as revoke or remove as required.

4. Empirical analysis & tests

The purpose of the experiments is to determine the effectiveness of the DED and PCS modules in controlling the database’s security in a test environment that comprises of
both normal and abnormal user activities. The database that the test is conducted on is an image that was duplicated from a pre-selected production database and the simulated activities that are derived from the database’s features that can capture the workload that occurs on a production database and replay it in another (Colle, Galanis, Buranawatanachoke, Papadomanolakis, & Wang, 2009). The database is part of an IT business system from a utility company whom permission had been obtained for this research. Due to sensitivity, all references to the business and company have been removed. The IT system is an enterprise data warehouse which has a high volume of activities that occur throughout the days. User login activities have been captured by the DB Replay (Colle et al., 2009) and replayed continuously to simulate the production environment. The experiments start with the initial batch of user activities that had been captured over 3 workdays starting from Monday. This form the initial raw dataset that the DED will work on. The dataset must be pre-processed before it can be used. Tasks such as removal of unrelated background information, missing value replacement are some of the prep works.

The main difference between our test and existing works (Khanuja & Adane, 2011; Partha et al., 2017; Schneider, 2003; Trejo et al., 2018) is the use of Reinforcement learning algorithm in conjunction with neural network and Apriori algorithm to detect database’s user activity anomalies, with another step to proactively remove suspicious users and their privileges from the database. Each database that belongs to any IT systems are unique and no two databases have the same pattern of usages. Our solution can learn its sequences of user activities within that DB local environment and able to detect and remove users with anomalous activities. Refer to Table 4 which has a fraction of the minded association rules using the algorithm, we use the most comprehensive rules with the most attributes for the sequence of user session data that are ranked with the highest lift as normal as highlighted in bold. This process reiterates for the other rules with different criteria until the data set is complete with the best set of data that reflect the current state of the database’s users’ activities that are regarded as common and normal.

For the NN test, a combination of epoch and batch size were tested against the dataset and the results are plotted in Figure 9. A variation of different batch size in steps of 10 were used while the number of the epoch was kept at 100 iterations. The accuracy of the NN’s
The prediction is measured in mean average percentage error (MAPE) and the result varied little when the NN was trained with different batch size.

The RL allows learning the best action for a given state. Given that NN may not be able to predict the best result for a given state as shown in Figure 6, it requires several iterations of the NN’s prediction until a local optimum result is achieved. The goal is to find the best global optimum result throughout the iteration and this is where RL excels – to learn global maxima. For the RL test, a batch of data (that comprised of a week’s worth of user session activity which was captured in the database before the start of this test) is used to generate the association rules within the DED module. The rule’s confidence and lift values form the basis for the neural network training data which informs which

---

**Table 4. Mined attributes for user session classification.**

| LHS | RHS | Support | Confid | Lift | count |
|-----|-----|---------|--------|------|-------|
| {USERNAME = SPLX2103,MACHINE = cbns1db02.common.local,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3), SCHEMANAME = SPLX2103} | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {HOST_NAME = cbns1db02.common.local,USERNAME = SPLX2103,OSUSER = qeedwhpr,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3)} | {INSTANCE_N = EEDWHPR1} | 0.117514 | 1 | 1.683495 | 9264 |
| {HOST_NAME = cbns1db02.common.local,USERNAME = SPLX2103,MACHINE = cbns1db02.common.local, PROGRAM = sp_opst_mt@cbns1db02.common.local (TNSV1-V3)},SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117514 | 1 | 1.683495 | 9264 |
| {USERNAME = SPLX2103,OSUSER = qeedwhpr, MACHINE = cbns1db02.common.local,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3), SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {HOST_NAME = cbns1db02.common.local,USERNAME = SPLX2103,OSUSER = qeedwhpr,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3),SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {USERNAME = SPLX2103,MACHINE = cbns1db02.common.local,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3), SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {USERNAME = SPLX2103,OSUSER = qeedwhpr,MACHINE = cbns1db02.common.local,SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {HOST_NAME = cbns1db02.common.local,USERNAME = SPLX2103,MACHINE = cbns1db02.common.local,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3),SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {HOST_NAME = cbns1db02.common.local,USERNAME = SPLX2103,OSUSER = qeedwhpr,MACHINE = cbns1db02.common.local, PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3),SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
| {USERNAME = SPLX2103,OSUSER = qeedwhpr,MACHINE = cbns1db02.common.local,PROGRAM = sp_opst_mt@cbns1db02.common.local(TNSV1-V3),SCHEMANAME = SPLX2103) | {INSTANCE_N = EEDWHPR1} | 0.117413 | 1 | 1.683495 | 9256 |
users’ sessions are normal. The DED builds up a knowledge-base and with it, it reads in subsequent new user sessions and determines their state of anomalies. For our test, we introduce a set of 30 new user sessions that don’t fit into the normal profile with an anomaly such as using an unknown program, logging from unregistered hostname, log in at an uncommon time of access including using rare OS user account like root. Figure 10 showed the progress of DED through 40 iterations. At the initial phase where the DED’s RL routine learns, it has learned a set of normal user activities, but it cannot predict the classification of those test user sessions accurately resulting in a series of false positives and negatives prediction. The erroneous result must be corrected manually by the IT administrator and add into the knowledgebase. This iterates until the 10th iteration where most of the test cases can be detected correctly. The result maintains at 99% accuracy with some exception where the new anomalous user session is introduced but at a much lower volume and frequency.

Both the DED and PCS modules are tested individually and separately before combining them together for the final integrated test. The DED is tested for its ability to detect user anomalies and the PCS is tested for its functionality to create and revoke user’s related objects with trial login to evaluate the logic of privilege allocation/de-allocation. The Final test is performed with the PCS and DED modules combined to evaluate their cross functionality together against several test case. The test is conducted on the R platform that has the capability of employing required libraries resources and invoke remote procedure calls to the database via the database client library. PCS’ trial test is deterministic as it is essentially procedure calls within the database and all the logics have been successfully tested. The actual test for the entire system revolved around the main module of anomaly detection, DED, to detect and initiate the PCS as a secondary function. As for the DED’s test, it has been observed that the number of false positives and negatives detection for the DED has substantially high at the beginning. But the error results start to diminish quickly when the initial phase of trial-and-errors ends in the DED’s reinforcement learning
agent and start on the next phase of learning the classification through both feedback loop of correction through loss function. Referring to Figure 10, the initial phase of the RL’s ability to predict accurate rewards did faltter for the first few iterations but with the knowledgebase that accumulated over each iteration, the volume of training set increase and that improve the RL agent’s NN prediction capability, which also equate to the improvement on accuracy of the anomaly classification.

The result is tabulated in Table 5 after processing with Privilege Control System. Based on the result, the basic integration is complete with no errors and the modules perform without any software failures. The result showed that there is still room for improvement on the prediction of the DED which fired off the PCS to deny the users. The cause for the misclassification was due to the flagging of anomalous sessions as normal by the IT administrators, human errors that can be prevented with better due diligence and proper procedure.

5. Discussion and Conclusion

This paper proposes a novel intelligent anomaly detection and privilege control solution for databases that use a series of machine learning methods including reinforcement learning with the Q-learning technique in conjunction with neural network and association rule mining. The goal is to improve the security of databases especially when there are increasing reports of mission-critical production databases that suffered

![Figure 10. DED's RL detection performance results.](image-url)

![Table 5. Final DED and PCS test results.](table-url)
information leakage from internal and external entities. Unlike other anomaly detection method that uses standard rules, we acknowledge that each database has their own unique users landscape or culture that cannot be governed by a one-size fit all rule policy. Our proposed system is adaptive toward each individual IT system that is unique on their own by learning the common itemset of users’ information and identify any user sessions that deviate from the common rules on these item-sets. It also has a feedback loop for IT administrator to correct and modify the state of detected user sessions to be either normal or anomalous to improve the overall detection knowledgebase. Another unique point of our system is the user denial functionality that is provided by PCS. While normal intruder denial or prevention systems can either detect or prevent anomalous sessions, it cannot be proactive to deny those sessions which have been classified as normal initially, has turned rogue. Our proposed system can meet to this gap of service for such requirement.

With this proposed system with DED and PCS modules, the database’s security landscape is consolidated into a central location without the need of spending a large amount for LDAP or Identity Access Management solution. It monitors the databases’ activity continuously for abnormal user activities and takes appropriate actions as programmed to drive the accessibility of the malicious user out of the systems cleanly and effectively. That minimizes potential risks that hackers could exploit. Users will have more difficulty to bypass the control and they will be caught in the act much faster as compared to passive alerting, the number of damages can be minimized drastically with PCS doing ring-fencing the database more rigidly, preventing cross schema confusion and implicit privileges that potentially have been missed. However, no system is totally secured but getting rid of all the pre-granted privileges on any databases is a significant step to secure the databases asset as well as attempting to monitor any user exploitation within the databases. This paper sets the foundation for the concept of DED/PCS.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Chee Keong Wee is doing his doctoral study in data science under the supervision of Dr Richi Nayak since January 2016. Chee Keong is also currently working as a senior database administrator with Energy Queensland and he holds multiple Oracle certified professional certifications. He has been working in the IT industry both as an IT support and consultant for the past 20 years.

Richi Nayak is the HDR director of Electrical Engineering and Computer Science faculty in the Queensland University of Technology and the leader of the Applied Data Mining Research group. Richi Nayak has contributed to the area of data mining and she holds multiple awards in researches and academic achievement. She is well regarded in the field of Data Mining and Artificial Intelligences and her papers have been cited frequently in the academic world.

References

Bertino, E., & Sandhu, R. (2005). Database security-concepts, approaches, and challenges. IEEE Transactions on Dependable and Secure Computing, 2(1), 2–19.
Buczak, A. L., & Guven, E. (2016). A Survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials, 18*(2), 1153–1176.

Colle, R., Galanis, L., Buranawatanachoke, S., Papadomanolakis, S., & Wang, Y. (2009). Oracle database replay. *Proceedings of the VLDB Endowment, 2*(2), 1542–1545.

Connolly, T. M., & Begg, C. E. (2005). *Database systems: a practical approach to design, implementation, and management*. 6th ed. Always learning. 1 online resource (1329 pages, A-5, B-12, C-6, D-6, E-18, R-17, FR-14, IN-24, CR-2).

Connolly, T. M., & Begg, C. E. (2006). A constructivist-based approach to teaching database analysis and design. *Journal of Information Systems Education, 17*(1), 43–51.

Gaedeke, M., Meinecke, J., & Nussbaumer, M. (2005). A modeling approach to federated identity and access management. in Special interest tracks and posters of the 14th international conference on World Wide Web. 2005. ACM.

Gaetjen, S. (2015). *Oracle database 12c security: Design and manage a secure enterprise database environment*. edited by D. Knox and W. Maroulis. New York: McGraw-Hill Education.

Hoffer, J., Ramesh, V., & Topi, H. (2015). *Modern database management* (pp. 155–205). New Jersey: Prentice Hall.

Khanuja, H. K., & Adane, D. (2011). Database security threats and challenges in database forensic: A survey. in Proceedings of 2011 International Conference on Advancements in Information Technology (AIT 2011), available at http://www.ipcsit.com/vol20/33-ICAIT2011-A4072.pdf. 2011

Kruk, S. R., et al. (2006). D-FOAF: Distributed identity management with access rights delegation. in Asian Semantic Web Conference. 2006. Springer.

Kuhn, D. (2013). *Pro Oracle database 12c administration* (2nd ed). Berkeley, CA: Apress. 1 online resource (xl, 714 p.).

Lampson, B. W. (2004). Computer security in the real world. *Computer, 37*(6), 37–46.

Liu, Y., & Chen, T.-Y. (2003). Security strategies of database Microsoft SQL server and its realization [J]. *Computer Engineering and Design, 1*, 019.

Liu, P., Jing, J., Luenam, P., Wang, Y., Li, L., & Ingsriswang, S. (2004). The design and implementation of a self-healing database system. *Journal of Intelligent Information Systems, 23*(3), 247–269.

Mellouk, A. (2011). *Emergence of intelligence through reinforcement learning with a neural network*. Advances in reinforcement learning (pp. 99–118). London: IntechOpen.

Partha, G., Bardhan, M., Chowdhury, N. R., & Phadikar, S. (2017). IDS using reinforcement learning Automata for Preserving security in cloud environment. *International Journal of Information System Modeling and Design (IJISMD, 8*(4), 21–37.

Performing Flashback Recovery. (2013). p. 395-442.

Pritchett, W. L., & De Smet, D. (2013). *Kali Linux Cookbook*. Packt Publishing Ltd.

Rahman, A., Ezeife, C. I., & Aggarwal, A. K. (2010). *Wifi miner: An online apriori-infrequent based wireless intrusion system, in knowledge Discovery from Sensor data* (pp. 76–93). Ontario: Springer.

Ramaswamy, C., & Sandhu, R. (1998). Role-based access control features in commercial database management systems. in Proc. 21st Nat’l Information Systems Security Conf. 1998. Citeseer.

Russell, S. J. N. (2016). Peter, Artificial intelligence: a modern approach.

Sandhu, R. S., & Samarati, P. (1994). Access control: Principle and practice. *IEEE Communications Magazine, 32*(9), 40–48.

Schneider, F. B. (2003). *Least privilege and more [computer security]*. *IEEE Security & Privacy, 1*(5), 55–59.

Sturrus, E., & Kulikova, O. (2016). *Identity and access Management* (pp. 208–218). Encyclopedia of Cloud Computing.

Szepesvari, C., Brachman, R., & Dietterich, T. G. (2010). *Algorithms for reinforcement learning* (pp. 49–51).

Tajbakhsh, A., Rahmati, M., & Mirzaei, A. (2009). Intrusion detection using fuzzy association rules. *Applied Soft Computing, 9*(2), 462–469.

Trejo, K. K., Clempner, J. B., & Poznyak, A. S. (2018). Adapting attackers and defenders patrolling strategies: A reinforcement learning approach for Stackelberg security games. *Journal of Computer and System Sciences, 95*, 35–54.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. in *AAAI*.