RBAC+: Protecting Web Databases with Access Control Mechanism

Archana Arudkar[1], Prof. Vimla Jethani[1]
[1]Mumbai University

ABSTRACT

With the wide adoption of Internet, security of web database is a key issue. In web-based applications, due to the use of n-tier architecture, the database server has no knowledge of the web application user and hence all authorization decisions are based upon execution of specific web application. Application server has full access privileges to delegate to the end user based upon the user requirement. The identity of the end user is hidden , subsequently database server fails to assign proper authorizations to the end user. Hence, current approaches to access control on databases do not fit for web databases because they are mostly based on individual user identities. To fill this security gap, the definition of application aware access control system is needed. In this paper, RBAC+ Model, an extension of NIST RBAC provides a application aware access control system to prevent attacks with the notion of application, application profile and sub-application session.

General Terms Preventing Attacks on Web databases by means of Access control mechanism.

Keywords Application Profile, Access control, Sub-application session.

1. INTRODUCTION

Web applications are extremely popular today, due to the simplicity of web brower and convenience of using web brower as a end user. The web applications have direct access to back end, called web databases, which contain sensitive and personal information of the end user. This information, if compromised can have a very serious impact on the organizations that deploy them and on the users who access them Thus, protecting data stored in web databases has become a strong need. Access control and views are primary means of attack prevention for databases. In case of web databases it is useless, because of the three or n-tier architecture, where real user's identity is hidden. So proper authorization cannot take place. In three tiers, all the requests are send by application server to the database server, so to fulfill the request, application server has given full privileges and the principle of minimal privilege is violated. It is impossible to authorized web application users with proper privileges at database level. Attackers can exploit these flaws to view sensitive data. Proper access control policies can not be implemented for databases. Therefore, web applications are exposed to many illegal access and attacks that are very hard to prevent and detect. Besides the famous SQL injection attacks there is one more kind of attack ,the Business Logic Violation attack for which satisfactory solutions are still lacking.

The central idea of RBAC+ is including the concepts of application, application profile and sub-application session when controlling the access to web databases. The application profile is necessary to track the user behavior throughout a whole session and mainly to prevent business logic violation attacks by enforcing access control.RBAC+ focuses on detection and prevention of malicious transactions by continuously monitoring the sequence of SQL statements issued by users. It monitors the malicious transactions and if identified cancels the transactions before it succeeds thus minimize the damage [1].

2. RELATED WORK

The problem of access control to database accessible over the web is very important. This

problem is known to web developers and security specialist. But little work has addressed it. Web databases are vulnerable to attacks like SQL injection, business logic violation and insider attack.Roichman and E.Gudes in [2] proposed a parameterized view with built-in access control mechanism to work with web applications to prevent intrusions. In this method, parameter is used to transfer the identity of user working with databases. So the requirement for this is the parameter should be difficulty to fake. One way to protect web databases from attack like SQL injection is to use ad-hoc tools which are used to detect the attack[3]. Another way is to use Intrusion Detection System (IDS)[4],[5].IDS is a good solution for detecting anomalous behaviors and thus play an important role in database security. IDS can not be used with proper internal access control and views to restrict the data access of web database.IDS focus on detecting attacks after the intruder has accessed the database. Another problem with IDS is that the detection phase of IDS contains the normal activities for anomaly detection purpose, which is only a subset of normal activities since the transaction learning depends on the utilization profile of the database. In many applications some transactions are performed only fort- night or at the end of the month. There is a coverage problem since it contains only frequently executed transactions. This gives false positives of anomaly detection based IDS.

One solution to this problem is to profile user behavior based on application logic. In web system, application interfaces are provided according to the business logic. This way it is possible to profile application features and reduced the risk of false alarm. By strengthening access control and continuously monitoring users, we can stopped many attacks from the access control stage.IDS can be used to detect attacks which are escaped from access control stage.Intrution detection without enforcing access control is not as efficient and effective.IDS alone can not protect databases from attacks.

3. RBAC MODEL

Role-Based Access Control (RBAC) is used for controlling access to computer resources. In RBAC, roles are created based on job functions of users. Permissions are assigned to roles based on the requirements of job functions. Users are made members of roles based on the job responsibilities and thereby gaining permissions assigned to the roles. This way in RBAC, users are granted permissions based on their roles, not on individual basis. This abstraction provided by role simplifies the management of permissions and thus helps to implement the principle of least privilege.

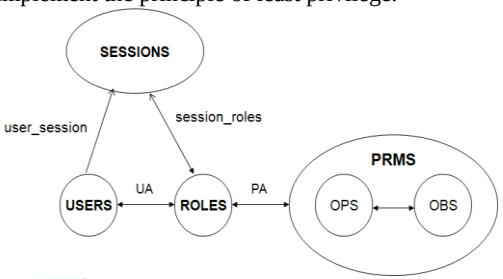

Fig 1 Core RBAC [1]

Core RBAC model is shown in figure 1 with following components.

The sets of USERS, ROLES, PRMS and SESSIONS represent the set of users, roles, permissions, sessions respectively.

- UA ⊆ Users x ROLES .The user-assignment relation that assign users to roles.
- Assign user: ROLES →2^{USERS} .The mapping from role to a set of users.
- PA ⊆ ROLES x PRMS.The permission assignment relation that assigns permissions to roles.
- Prms Assignment: ROLES →2^{PRMS} .The mapping of a role into a set of permissions.
- Session User: SESSIONS→USERS.The mapping from a session to a user.
- Session Role: SESSIONS→2^{ROLES}. The mapping from a session to set of roles.[1]

4. OVERVIEW OF THE APPROACH

In web system, access to data occurs through several layers, starting with end users, web server, application server and then databases. It is difficult to grant permissions to the user since end users identity is hidden. All the requests of end users are submitted by application server. Another problem is single user's transactions cannot be trace to seek signs of anomalous behavior. The solution to these problems is RBAC$^+$, an extension of NIST RBAC, able to detect malicious transaction and stop the attack before it succeeds.

Assuming that the database management system (DBMS) has an RBAC model in place, the concept of the approach is as follows. Here application profile represents, an execution path, a sequence of SQL queries for the execution of a task.

Necessary permissions are given to the application for execution and set of roles are authorized to database users(DBU),for each pair of(application,DBU) the subset of roles are activated in a user's session, called sub-application session. A sub-application session contains only permissions needed to execute a created task and take advantage of RBAC asset such as least privilege and separation of duty. A sub-application session allows DBMS to distinguish between web users working with database, thus solving the first major problem of fine grained authorization at the database level. It will also allow distinguish the requests of different web users having same database session, thus solving second problem of user's session traceability for web applications.

When user logs in, the SQL queries that he submits are associated with database session, an application and the database user that issued them. All the queries of a sub-application session must match an application execution path else access is denied because the transaction is considered as malicious and rolled back. Privileges are limited only to legitimate actions. The important of this solution is that it enforces access control based on business application logic rather than primitive reads and writes. A users can access and manipulate data depends on the application function they execute. This drastically reducing attack like business logic violation attack. Take example of online shopping application. The process involves following stages: 1) Browse the product list and add items to the basket.2) Finalize the order.3) Submit credit card details.4) Enter delivery information.

When an employee wants to attack enterprise resources, and if he submits SQL injection attack.SQL injection is entirely fail or at least its effect is very limited because the user's database privileges are limited to legal actions only.

If the intruder submits insert statement into Orders table without submitting an insert into Credit card table, then he buys goods without paying. This violates the business rule and it can be detected at session level since each statement is valid statement. Database cannot stop such attack because authorization is on the basis of user's identity not on the basis of business logic of an organization. The developers assume that users will follow the stages in sequencer and navigational links and form interfaces are provided according to sequence of stages to the web browser. But navigational flow is under the control of user and may access the stages in any sequence.

When multistage functions are accepted in out of expected sequence, it is common to encounter a variety of anomalous conditions within the application, such as variables with null or uninitialized values, a partially defined or inconsistent state, and other unpredictable behavior. In this situation application may return error messages and debug output, which can be used to better understand its internal working and thereby fine tune the current or a different attack. Sometimes the application may get into a state entirely unanticipated by developers, which may lead by serious security flaws [7].

5. THE CORE RBAC$^+$ MODEL

The RBAC+ model has major components such as application, sub-application session and application profile. The model is shown in the fig.2 below. This graphical representation is adopted from NIST RBAC model. In this APPS,

AP, SASES represents sets of applications, application profiles and sub-application session respectively.

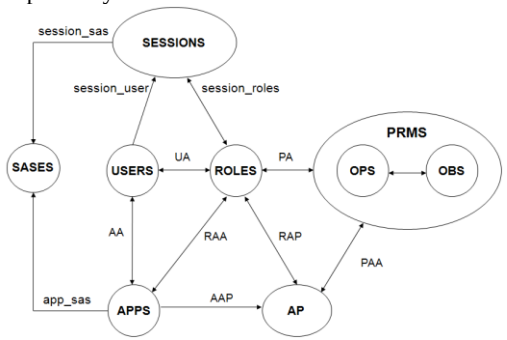

Fig 2 Core RBAC^{+} Model [1]

5.1 Application profile

An application can be executed in many possible ways which is nothing but an application profile. An application profile can be defined as sequences of selects, inserts, updates and deletes to perform a specific task of an application. An application profile is a sequence of nodes and edges between the nodes. Each node represents a SQL statement. Web application by using DML operations interacts with databases. Therefore application profiles are built by analyzing application code.
From the model, we can define:

- AAP: APPS→AP, the mapping of an application onto its corresponding application profiles. Formally, APPS_profiles (apps) = {apЄAP|(ap,apps) Є AAP}.
- RAP ⊆ AP x ROLES, a many to many mapping of role to application profile assignment relation.
- AP_roles (ap) = {r Є ROLES| (r, ap) Є RAP} [1].

5.2 Sub-application session

An application session is set of all transactions of all its user. A sub-application session (SASES) is subset of transaction of one user. Formally,

- app_sas: APPS → 2^{SASES} .The mapping of an application onto a set of sub-application session.
- Ses_sas: SESSION→2^{SASES}.The mapping of a session onto a set of sub-application sessions [1].

5.3 Permissions

Permissions are associated with roles and application profiles. Applications are associated with the appropriate roles based on the set of permissions assigned to application profiles. This is given by two functions: PAA and RAA.
PRMS is defined as PRMS= $2^{OPS \times OBJ}$

- PAA ⊆ PRMS x AP, a many to many mapping of Permission to Application profile assignment relation.
- AP_perms: AP → 2^{PRMS}, the mapping of an application profile onto set of permissions. Formally, AP_perms (ap) = {pЄPRMS| (p, ap) ЄPAA}.
- RAA ⊆ ROLES x APPS, a many to many mapping of Role to application assignment relation.
- APPS_roles: APPS → ROLES, the mapping of an application to a set of roles. Formally, APPS_roles (app) ={r Є ROLES | (r, app) ЄRAA}.[1]

5.4 Users

Each user is aassociated with a set of applications .Given set of users, the following relations are defined:

- AA ⊆ APPS x USERS, a many to many mapping of application to user assignment relation.
- USER_AssignmentApps: USERS → 2^{APPS}, the mapping of a set of applications. Formally, USER_AssignApps(u)={u Є USERS | (app,u) Є AA}.[1]

5.5 Sessions

When a user logs in, a new session is activated and a set of roles are selected .For a given session s, three functions are defined:session_user (s) corresponds to the user of the session; session_roles (s) corresponds to the roles activated in a session;session_applications (s) corresponds to the applications using this session; Given a session and application two function are

defined:avail_app_roles (s,app) corresponds to roles available for an application.; avail_app_perms (s,app) corresponds to the permissions available to an application in a session. Formally defined as:

- Session_user(s): SESSION → USERS, the mapping from a session s to the user of s.
- Session_roles(s): SESSION → 2^{ROLES}, the mapping of session s onto a set of roles. Formally: session_roles(s) ⊆ {r ∈ ROLES | (session_user(s), r) ∈ UA}.
- Session_application: SESSION → 2^{APPS}, the mapping of session onto a set of applications.
- avail_app_roles : (SESSION,APPS) → 2^{ROLES}, the roles.Formally,avail_app_roles(s,app) ⊆ {r ∈ ROLES | r = session_roles(s) ∩ app_roles(app)}
- avail_app_perms: (SESSION, APPS) → 2^{PRMS}, the permissions available to an application in a session.Formally,avail_app_roles(s,app)=$U_{r \in avail\_app\_roles(s,app)}$ assigned_permissions(r)[1].

6. BUILDING APPLICATION PROFILE

In database environment, transactions are fixed till the application is not change. For example, in an online banking application, users can only perform the operation like withdraw money and check balance. No other operations are allowed to the end users. An application profile is nothing but a sequence of DML operations related to each other in terms of business logic. Build the application profiles and use it for access control. The application profiles can be build by using following three ways:

- Manual profiling can be used to build application profile if the transaction is not large.
- Running application test. By using testing tools can generate all application functionalities.
- By analyzing code of application program can generate application profile. Because application interacts with databases using DML commands.

Example:

- Consider a web application of brokerage firm having following roles:
- Guest users can go through security details and can read market news.
- Customers can submit trade request on her account.
- Brokers can submit trades to the market on behalf of customers.
- Newsman can update news of the market.
- Markets a group of users can actually submit the transaction and updates the status of the transaction.

Each role has different permissions at database levels: Newsman has insert permission on News table, but all others have select permissions. Customers have insert, update, and delete permissions on Trade table. Brokers have update permission and all other roles have no permissions. Node 1 represents select, Security_details; Node 2 represents select, News; Node 3 for insert, Trade; Node 4 for update, Trade; Node 5 for delete, Trade.

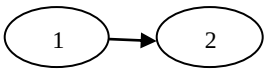

Fig 3 Application profile for Guest user

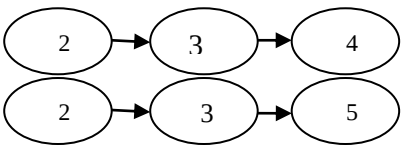

Fig 4 Application profile for Customers

7. ACCESS CONTROL

The session roles are the roles that are assign to the user of the session. So the application has set of roles that the database user is authorized for. A user session contains many sessions belonging to the same or different applications. If it is of same application then the roles assigned to user is same as that of application and if it belongs to different application then needs to activate subset of roles. To enable the roles, PAA function accepts as a input the set of roles assigned to users and

permissions covering the application profiles of an application and tries to find out optimal set of roles covering application profiles permission and all role constraints within the system. To activate set of roles in a session covering requested permissions, satisfying role constraints that prevent activation of conflicting roles in a session and following principal of least privileges a Role Mapping algorithm is given which is inspired by [6].In order to improve the performance, roles with extra permissions than requested permissions are removed in the beginning of the search.

7.1 Access Control Policies

Application profiles are used to detect the unauthorized SQL statements. which are considered as invalid based on the application logic. The authorization control function is defined as follows:

An access request ar is a tuple *ar* = < U.is, app, p, o> Є *USERS x SASES x APPS x OPS x OBJ* [1].ar can be satisfied if (p, o) Є avail_app_prms(s, a) and is Є session_sas(s).The above function is repeated as many permissions as the SQL query requires permissions to be executed. To protect the information stored in databases which are accessed by web users, the access control policies must be flexible enough. Two access control policies can be used.

7.2 Policy 1

This policy monitors all transactions of a user. If a transaction is a new transaction, tool searches all the application profiles starting with first requested command of the newly entered transaction. If the application profile starting with requested command is found, it will be considered as candidate application profile. The next command is matched with the command in candidate profile. This process is repeated till the end of the transaction. If no candidate profile is found for the transaction, it is considered as malicious transaction and rolled back.

7.3 Policy 2

Under this policy, all the SQL statements submitted by the user are stored as user context. Access is granted to the user until she submits critical point. Here critical point is SQL statements which change the state of the database i.e. (insert, delete, update)

8. CONCLUSION

RBAC^{+}, an extension of RBAC model, suggests access control mechanisms for RBAC implemented web databases. This model not only detects the attacks but also stop the attacks when they are detected and thus minimized the losses caused by the attacks. Access control policies can be implemented by using PL/SQL language. The primary requirement for this approach is the source code of web application to build application profile. The defense-in-depth technique means the multilayer system, can be implemented. In this, first layer and second layer are used to prevent and detect the attacks respectively.

9. REFERENCES

[1] Ahlem Bouchahd A.,Nhan le Thanh Adel bouhoula,Faten Labbene,"Enforcing access control to web databases",2010 10th IEEE International Conference on Computer and Information Technology(CIT 2010).

[2] Roichman and E.Gudes,"Fine-grained access control to web databases", in ACMAT 07 : Proceedings of the 12th ACM Symposium on Access Control Models and Technologies,AC,2007,pp.31-40.

[3] W. G. Halfond, J. Viegas and A. Orso."A classification of sql-injection attacks and countermeasures", in Proceedings of the IEEE International Symposium on Secure Software Engineering,Arlington,VA,USA,2006.

[4] S. Y. Lee, L. Low and P. Y. Wong," Learning fingerprints for a database intrusion detection system", in ESORICS 02: Proceedings of the 7th European Symposium on Research in computer security.London, UK: Springer-Verlag.2002, pp.64-280.

[5] E. Bertino,A. Kamra,E. Terzi and A. Vakali."Intrusion detection in rbac-administered databases", in ACSAC 05: Proceedings of the 21st Annual Computer Security Applications Conference.

Washington, DC, USA: IEEE Computer Society, 2005, pp.170-182.

[6] G. T. Wickramaarachchi.W. H. Qardaji and N. Li,"An efficient framework for user authorization queries in rbac systems", in SACMAT 09: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. New York, NY, USA: ACM, 2009, pp.23-32.

[7] Faisal Nabi,"Designing a Framework Method for Secure Business Application Logic Integrity in E-Commerce Systems", International Journal of Network Security, Vol.12, No.1, PP.29-41, Jan.2011.