ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 1, pp: (43-46), Month: January - March 2022, Available at: www.researchpublish.com

REAL-TIME DATABASE SECURITY MONITORING FRAMEWORK USING NATIVE DATABASE AUDITING

[1]Yahya H. Dossary, [2]Hamad A. Maghlouth

[1,2] Dhahran, Saudi Arabia

Authors Email: yhdossary@yahoo.com, maghham@yahoo.com

***Abstract*: The are several commercial real-time database security monitoring platforms available in the market. These commercial platforms are mostly available for major databases. They are not readily available for new commercial databases and open source databases. This paper proposes a framework to establish real-time security monitoring for databases that are not supported by any of the commercial platforms. The framework implements selective native database auditing functionality that forwards audit entries to a Security Information and Event Management (SIEM) via the operating system log forwarding daemon. The framework focuses primarily on protecting the database from insider threats. The solution can be used as permanent or an interim workaround until the database is supported by a commercial platform.**

***Keywords*: database auditing, real-time security monitoring.**

I. INTRODUCTION

Database systems are mission critical components of IT business applications. The Database stores and processes business data which is one of the most valuable business assets. The majority of IT infrastructure components that power business applications can be rebuilt after a major outage. However, if the business data is damaged or corrupted it is extremely hard to rebuild it. Therefore, it is mandatory to secure databases and to establish a framework that enables real-time database security monitoring [1].

The are several commercial real-time database security monitoring platforms available in the market. These commercial platforms are mostly available for major databases. These platforms are not readily available for new commercial or open source databases and if they are available significant license cost is usually associated with their deployment.

Database administration and security teams often face the challenge of complying with external and internal security controls mandating real-time security monitoring and the lack of off-the-shelf solutions that simplifies complying with regulatory or corporate standards [2].

This paper proposes a framework that is developed with multiple threat cases to establish real-time security monitoring for newly introduced database that is not supported by any of the commercial database security platforms. The framework is developed using native database auditing functionality that forwards audit entries to a SIEM system (i.e. HPE ArcSight ESM) on-the-fly via the operating system log forwarding daemon (ex. Linux Syslog daemon). The audit entries are generated by multiple selective audit policies. The solution covers audit generation, collection, preservation, and analysis.

The SIEM system is used to develop and run the threat cases' logic. The threat cases output is forwarded to a central security monitoring console to provide 24X7 real-time security monitoring. The SQL statements listed with each sample threat case definition presents the basic logic used in each threat case. Similar SQL statements can be used to develop the same threat cases directly on the database audit table if a commercial SIEM is not available to the organization.

An SIEM system such as Splunk can also be integrated into the framework to preserve the audit records for an extended period for time and to act as the solution forensics database for reactive analysis of database activities.

The following parts of this paper are organized as follows: section 2 will present the architecture of an illustration application that is monitored with the seven sample threat cases. Section 3 presents the layout of the audit trail that is generated by the database and the audit polices that were created. The details of the seven threat cases will be described in detail in section 4. We will conclude the paper in section 5.

II. APPLICATION ARCHITECTURE

The database to be monitored is part of a 3-tier application, MyHR, that connects with a generic user HR to the backend database HRPRD. The MyHR application connections originate from a program called HRlib. The MyHR application is powered by two application servers HR1 and HR2 that connect to the backend database hosted in the server HRDB1. The HR user is the only user in the HRPRD database that is allowed to perform data definition language (DDL), Data Manipulation. Language (DML), and Data Access operations. Figure 1 illustrates the architecture of the proposed framework.

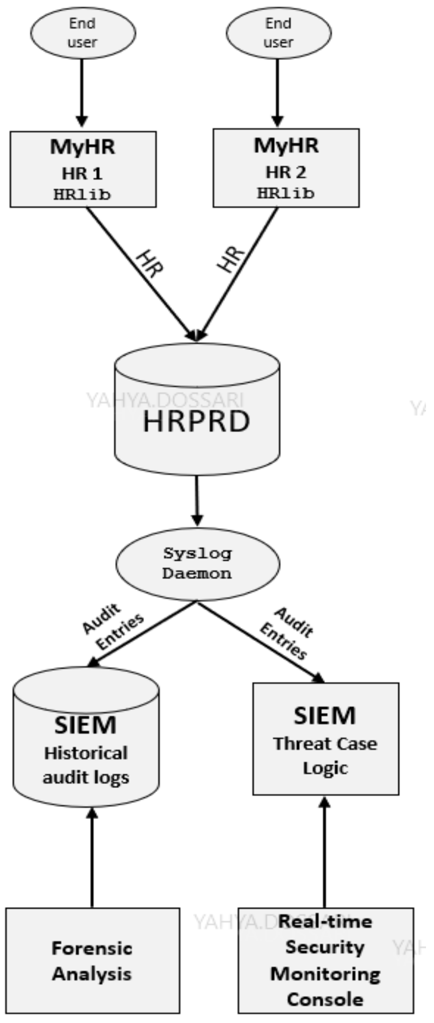

Fig. 1: MyHR Application and Solution Architecture

III. AUDIT POLICIES AND AUDIT TRAILS

Multiple audit policies are created as part of this framework. The audit polices define the actions to be audited and the conditions that triggers the audit events to be written to the database audit trail. Tables 1 lists the audit polices of the framework. The database generates an audit trail with the layout shown in Figure 2. The audit trail used in the sample use cases forwards the audit entries to operating system log forwarding daemon but a database audit table can be used instead.

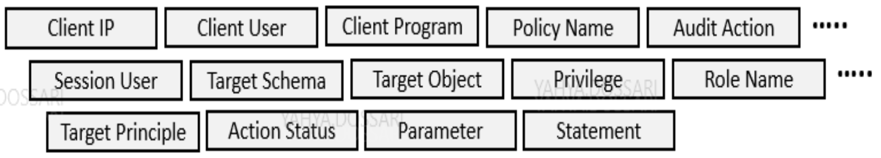

Fig. 2: MyHR Application and Solution Architecture

Table I: List of Selective Audit Policies

POLICY	DESCRIPTION
AUDIT_OPER_AUD_PLCY	Captures audit policy operations such enabling and disabling auditing
CONNECT_OPER_AUD_PLCY	Captures connection operations entries such as connection user, program and client
IMPERSON_OPER_AUD_PLCY	Captures operations against myhr.users table
PRIV_ESCLATE_AUD_PLCY	Capture privilege and role assignment operations
DATA_EXP_AUD_PLCY	Captures data export operations
DATA_DENCRYPT_AUD_PLCY	Carputers data volume encryption operations

IV. THE THREAT CASES

A. Audit Operations Threat Case

The first actions that are audited are operations related to auditing in particular disabling of audit policy as an offender does not want to leave any trace of malicious operations. A threat case is developed on top of the audit trail to capture the client details in case the AUDIT_OPER_AUD_PLCY is triggered with a disable operation. The following SQL statement illustrate the basic logic of the threat case developed in SIEM:

SELECT client_ip, client_user, Statement

FROM audit_trail

WHERE policy_name= 'AUDIT_OPER_AUD_PLCY' **AND** Statement like 'ALTER DATABASE DISABLE AUDIT POLICY%'

B. Unauthorized Client Connection Threat Case

The MyHR application connects to the database with a generic database user HR. The HR user is the most privileged user in the database because it has access to business data. The MyHR application servers are the only clients that should be allowed to connect to the database with the HR user. Any connection to the HRDB database with HR user from client other than the two application servers should raise a flag. The below SQL statements illustrates the basic logic of the threat case. The client_program filter adds another security layer to prevent the DBA from login to HRDB from a SQL client that is installed on the application servers.

SELECT client_ip, client_user, client_program

FROM audit_trail

WHERE policy_name= 'CONNECT_OPER_AUD_PLCY' **AND** session_user='HR' **AND** client_IP **not in** ('HR1', 'HR2') **AND** client_program <> 'HRlib';

C. Failed Login Attempts Threat Case

Privileged users' connections should be tracked and multiple failed logon attempts should be reported. The following SQL statements illustrates the basic logic of this threat case:

SELECT client_ip, client_user, client_program

FROM audit_trail

WHERE policy_name='CONNECT_OPER_AUD_PLCY' **AND** session_user in ('HR', …) **AND** Audit Action='CONNECT' Action_Status= 'FAILED'

D. Privilege Escalation Threat Case

Database administration best practices require a form of segregation of duties to be enforced. DBAs Should not be allowed to execute any operation that would escalate their privileges.These should be executed by a security administrator (SECURITY_ADMIN_n). A threat case should be developed to monitor ALTER USER and GRANT operations. The following SQL statements illustrates the basic logic of the threat case:

```
SELECT client_ip, client_user,
FROM audit_trail
WHERE policy_name='PRIV_ESCLATE_AUD_PLCY' AND session_user not in ('SECURITY_ADMIN_1',
'SECURITY_ADMIN_2');
```

E. Data Export Threat Case

The database administrator may not always access the database stored in database tables directly. The DBA can export data outside the database (ex. *.csv file) and then access the data using other programs such as a spreadsheet. A threat case should be developed to monitor export operations. The following SQL statements illustrates the basic logic of the threat case:

```
SELECT client_ip, client_user,
FROM audit_trail
WHERE policy_name='DATA_EXP_AUD_PLCY' AND Audit_Action='EXPORT';
```

F. Data Volumes Decryption Threat Case

Corporate policies often mandate that database data volumes to be encrypted at reset to prevent the database data from being accessed using operating system utilities. A Data Volume Decryption threat case was created to monitor decryption operations. The following SQL statements illustrates the basic logic of the threat case:

```
SELECT client_ip, client_user,
FROM audit_trail
WHERE policy_name='DATA_DENCRYPT_AUD_PLCY' AND Audit_Action='DECRYPTE';
```

V. CONCLUSIONS

Insider threat is a major risk that organizations must address to protect their data. Native database auditing is a simple and easy to deploy tool to mitigate this risk especially when a commercial security tool is not available. This paper presented seven sample threat cases that were implemented to establish real-time security monitoring of insider activities.In addition to providing real-time monitoring of insider activities, the selective auditing entries can be used to establish accountability for actions, deter DBAs from inappropriate actions, investigate suspicious activities, detect problems with access control implementation, and generating an audit record for audit and compliance personnel [2]. The same native auditing framework can be extended to add additional threat cases that are specific to other organizations.

REFERENCES

[1] Elisa Bertino, Fellow, IEEE, and Ravi Sandhu, Fellow, IEEE "Database Security—Concepts, Approaches, and Challenges" IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 1, January-March 2005.

[2] Ben Natan, Ron, "Implementing Database Security and Auditing", Elsevier Digital Press, 2005.