Realtime publishers

The Essentials Series: Role of Database Activity Monitoring in Database Security

Mitigating Risks and Monitoring Activity for Database Security

sponsored by

by Dan Sullivan

Copyright Statement

Mitigating Risks and Monitoring Activity for Database Security

As with other areas of information security, database security practices are driven by the need to mitigate risks. Conducting database discovery processes and vulnerability assessments are the first steps to implementing a comprehensive database security strategy. Once those two steps are done (as outlined in the first two articles in this Essential Series), it is time to move on to assessment and mitigation stages. In this, the final article of this series, we outline fundamental steps in assessment and mitigation and then consider the role of database monitoring for ongoing security.

Post-Discovery and Post-Vulnerability Scanning Assessment

After discovery and vulnerability scanning operations, we have the information necessary to understand what databases are in an organization and what their conditions are from a security perspective. Now it is time to act on that newly acquired information.

The first step is to determine which database instances are needed. Those that are not needed should be eliminated. In some cases, multiple databases can be replaced by a single instance. Consolidating databases under a controlled number of instances may reduce workload and simplify security and risk management processes. For example, rather than several developers each working with their own database instance, they could use a single group development database. In another example, two databases each supporting a small number of application-specific schemas could be combined into a single database instance supporting all application-specific instances. Each application schema could be isolated from the other using access controls so that the benefits of having separate databases remain without the overhead of having to maintain multiple database instances.

For each of the databases that will continue to be used, the next step is to ensure proper controls are in place for the category of data stored in the database. Development databases that do not contain real customer data require fewer controls than production databases with private and confidential data. Security controls should include access measures that:

- Limit the ability to read, update, and delete according to a user's role in the organization and that person's business responsibilities
- Include proper file system protections on database files to prevent tampering with storage files and other files needed for databases to function
- Incorporate host intrusion prevention systems on high-valued databases to detect unauthorized activity on the server
- Ensure that gateway controls, such as firewalls, control access to the database instance

As a general rule, patching a security vulnerability is preferable to leaving the vulnerability in place. Unfortunately, it is not always feasible to apply patches. Sometimes long-lived enterprise applications have been augmented with custom code that would break if a particular patch were to be applied. In other cases, patches may not be available for older versions of the database and upgrading to newer versions is not a practical option. We should remember that best practices are general rules that must be adapted to the constraints and limitations of real-world implementations. In some cases, we may need to devise mitigation strategies when patching is not an option. These can include tighter firewall controls, additional database activity monitoring, and the use of intrusion prevention systems.

Mitigating Database Server Risks

The operating system (OS) running on the database server, or servers in the case of a cluster-based database, should be properly patched and hardened. That is, unnecessary services should be shut down on the OS. Just as optional components in the database management system can harbor avoidable vulnerabilities, unneeded OS services can expose the database unnecessarily to vulnerabilities.

Policy Reviews

In addition to patching, one should review and revise security policies based on the findings of the vulnerability assessment. Three types of policies, in particular, should be reviewed:

- Access control policy
- Separation of duties policy
- Rotation of duties policy

Access control policies define rules for authentication and authorization practices. They should cover authentication issues such as what forms of authentication are required on different types of databases. For example, username and password authentication may be sufficient for some databases but two factor authentication, such as the need for a password and a smart card, is required for others. Access control policies should also define rules governing privileged accounts in the database, access to file system directories containing database files and code, and any controls related to physical access to database servers.

Separation of duties is an increasingly important issue. The basic principle is that any critical function should require more than one person to complete. The principle is well known in financial management where it is used to reduce the chance of fraud. An example of separation of duties in the financial area is the person responsible for generating invoices cannot be the same person processing payments for those invoices. In database management, roles such as database administrator (DBA), database developer, systems manager, and network administrator should be assigned to different members of the staff. Separation of duties is especially important from an audit and compliance perspective; organizations without sufficient separation of duties may be identified as such during an audit.

More on Separation of Duties

See the Information Systems Audit and Control Association's (ISACA) Segregation of Duties Matrix for other examples of duties that should not be combined into a single position at http://www.isaca.org/AMTemplate.cfm?Section=CISA1&Template=/ContentManagement/ContentDisplay.cfm&ContentID=40835.

A related principle is rotation of duties in which different people perform a task at different times. For example, one person may be DBA on a database for 6 months and then rotate to manage another database. The idea behind rotation of duties is that errors or malicious activity by an insider may be detected if another, non-colluding employee assumes that person's responsibilities for some period of time.

Demonstrating Compliance

In addition to having appropriate security controls in place, we need to be able to prove we have them in place as is required by many regulations. This requirement includes demonstrating that alerts are in place, management reports are available, and policy documents are up to date. Policies and methods for enforcing those policies should all be mutually supporting from a documentation perspective. Again, special attention should be placed on being able to demonstrate separation of duties is practiced.

Next on the list of important functions related to database security is the need for ongoing security controls such as those provided by database activity monitoring.

Ongoing Security Controls and Database Activity Monitoring

Having proceeded methodically from discovery to vulnerability scanning and through to assessment and mitigation, a database management team would have an organization's database properly configured and under formal management and would be enforcing policies appropriate for the category of data maintained as well as using other necessary security controls. It is an important milestone to reach that point, but it is not the end of the process. Ongoing security requires database activity monitoring. To better understand this control, let's examine both database activity monitoring functions and implementation issues.

Database Activity Monitoring Functions

Database activity monitoring provides a number of functions beginning with monitoring database transactions, including both data definition and data manipulation transactions. Data definition activity relates to creating tables, views, indexes, integrity constraints, and other structures as well as dropping these same artifacts. Data manipulations include inserting, updating, and deleting data from the database. Out-of-the-ordinary data manipulation and data definition activity can indicate malicious activity, such as dropping integrity constraints to hide improper changes to data or selecting a large number of rows from a table that contains customers' personal financial data.

In some cases, just logging information about an unusual event is enough as long as we have details about the who, what, when, where, and how of the event. In other cases, we may want to block the unauthorized transaction. Ideally, blocking is driven by policies defined according to the types of authorized operations on databases. Some examples of transactions that warrant blocking include:

- Dropping a table from a production system during normal operating hours is probably done in error and should be blocked
- Running a large extraction, transformation, and load operation on a data warehouse outside of the normally schedule time
- Long-running queries that consume too much CPU

  Note

  This can occur in production transaction processing systems that support ad hoc queries and allow users to query, for example, all customer transactions for the past year grouped by the type of products purchased and the total revenue from each order. There may be a legitimate business need for this but such queries should be run on a data warehouse tuned for such queries, not transaction processing systems tuned for interactive and high-volume transactions.

- Queries that attempt to retrieve too much data may be part of a data breach and should be blocked while the legitimate need for that data is verified. For example, a user may have reason to retrieve one sensitive record at a time but not all sensitive records at once.

In addition to data-oriented operations such as blocking, database activity monitoring systems should support functions such as:

- Tracking activities by user to help identify abnormal activities on the database—This is especially important for tracking activities performed by DBAs, accounts used in shared pool configurations such as Web applications, and accounts with credentials hard coded into scripts.
- Generating realtime alerts when a policy violation occurs in order to take immediate action to block unauthorized activity—The information generated in these alerts can also help demonstrate compliance procedures are in place.
- Implementing application firewall services to prevent unauthorized requests—These can help mitigate the risk of vulnerabilities in applications that cannot be corrected. Application firewalls prevent unauthorized requests, such as queries for a large number of sensitive records, from being executed on the database. This is different from blocking, which occurs after an operation has begun.
- Addressing common implementation issues such as using a single database connection for multiple application users—This process, known as *connection pooling,* has required a tradeoff between security and performance in the past. It is not always obvious which application user is utilizing a shared connection unless developers implement extra code to track application users with transactions against a connection. There is such a performance gain with not having to set up and tear down database connections for every separate user or transaction that connection pooling is widely used; the cost, however, is more difficulty in holding users accountable for their activities on the database.

With regards to implementation options, database activity monitoring can be performed using either a network appliance model, an agent-based model, or a combination of the two. Network appliances have a number of advantages. They can apply controls to multiple vendor databases, enforce separation of duties, and offload security operations for the database server. Agent-based methods are well adapted for local blocking and monitoring. Although the database activity monitoring market is young and still evolving, we are likely to see more use of a hybrid model in which the network appliance and agent-based systems are employed together in order to realize the benefits of each.

Benefits of Activity Monitoring

From a broader business perspective, three of the key benefits of database activity monitoring include:

- Improved security monitoring
- Improved ability to demonstrate compliance and generate compliance reports
- More effective monitoring of database and application integration points

Database activity monitoring helps remove some of the unwelcome tradeoffs we have had to live with in the past, such as the tradeoff between performance and the ability to monitor application users' activity in the database when pooled connections are used. As database activity monitoring technology improves, we will likely see additional developments that reduce the burden on application developers to create custom security and auditing solutions or live with the limitations of existing database application security controls.

Summary

Database activity monitoring is a promising technology for improving database security. To maximize the benefit of this technology, we must implement it properly and that, in turn, requires we undertake a methodical process of database discovery, vulnerability scanning, and assessment and mitigation before engaging a database activity monitoring system. Taking the time to properly implement and maintain database activity monitoring can yield benefits long after that initial implementation.