Realtime publishers

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

Database Assessment and Management in Database Security

sponsored by

IMPERVA®

by Dan Sullivan

Copyright Statement

Database Assessment and Management in Database Security

Database assessment is a process for determining weaknesses in database security. The first article of this Essential Series discussed how to use database discovery to identify databases within an organization—including those that have been informally managed and not under proper controls. Regardless of the past history of a database instance as either formally or informally managed, its security status should be assessed. Vulnerabilities can creep into even well-managed databases unless those systems are reviewed and assessed for common weaknesses. This article examines three topics in the area of database assessment:

- Vulnerabilities in database management systems
- Vulnerabilities in data management practices
- Vulnerability assessment practices

The article concludes with a brief discussion of how to prioritize vulnerabilities and plan for mitigating them.

Vulnerabilities in Database Management Systems

An almost axiomatic principle of information security is that complex systems have vulnerabilities. Database management systems are some of the most complex applications running in many businesses. Within enterprise-scale database systems, we find millions of lines of code, multiple components, and modules with different interfaces using various communication layers. Distributed databases and highly scalable databases running on clusters include additional layers of complexity. To put it bluntly, the code underlying modern database management systems is so complex there is no shortage of opportunities for introducing vulnerabilities. Compounding the potential for vulnerabilities in database software, vulnerabilities can arise from errors in configuration and inappropriate management practices.

The types of vulnerabilities in database management systems that should be considered in a vulnerability assessment include:

- Privilege abuse
- Database management software vulnerabilities
- Misconfiguration vulnerabilities
- Application vulnerabilities

Vulnerabilities from any one of these classes can put data at risk, and it would not be surprising to find multiple types of vulnerabilities within a single database.

Privilege Abuse

One of the biggest vulnerabilities arises from how we manage databases, not from the weaknesses in database systems. The problem is privilege abuse and it stems from having too many user accounts and granting too many privileges to those accounts. In many ways, the same difficulties we have with user provisioning at the network level occur at the database level. (Which is yet another reason database administrators (DBAs) and network managers need to work together on security; see the first article in this Essential Series for more on that topic.) A common practice is to use shared database accounts for applications so that every application user does not need a corresponding database account. This setup alleviates some provisioning/de-provisioning workload, but these shared accounts should still be monitored to ensure they are not abused. To address this problem, employ the three-step process described in the following paragraphs.

Audit existing accounts and determine which should be active. Accounts assigned to individuals should be active only if that person is still with the organization and in a role that requires access to the database. Accounts used for programmatic access should be associated with active applications. Those applications should be reviewed to understand the types of operations performed on the database; that information will guide how to assign privileges to the account. Also, determine whether scripts hard code database credentials and, if so, ensure adequate protections are in place to prevent disclosure of those credentials.

Adjust authorizations to appropriate levels. Authorizations should be granted based on a well-defined business requirement. Also, create roles within the database that correspond to business functions. Doing so can help avoid cases in which a user has need for authorization A but is assigned to a role with authorization A, B, and C because that role has the least privileges of all existing roles that still meet the business need.

Identify high-risk accounts and monitor appropriately. DBAs have long had complete access to databases. This access includes not only the data in application schemas but also logs and metadata about the structure of databases. The integrity of audit and log information must be preserved to ensure the integrity of the entire database. Even in cases in which DBAs may have the ability to make changes, knowing that such changes are monitored may dissuade inappropriate activity.

In the long term, privilege abuse may be controlled by aligning network user provisioning with database application provisioning when possible, implementing a regular account and privilege review, and logging alerts when authorizations are changed or accounts are created.

Database Management Software Vulnerabilities

As mentioned earlier, database software is complex and we should expect it to have security flaws. Some examples of database vulnerabilities include:

- A vulnerability in the Oracle database listener allows attackers to compromise the availability of database services (CVE-2009-0991)
- Several versions of Microsoft SQL Server do not initialize memory pages when reallocating memory, allowing operators to obtain access to potentially sensitive information (CVE-2008-0085)
- A vulnerability in an IBM DB2 database add-in for Microsoft Visual Studio and Visual Studio .NET allows authenticated users to execute arbitrary code (CVE-2008-3852)

We can also expect vendors to patch those flaws, which they do—but sometimes at a slower pace than some would like. However, there are some who would argue that patches should not be applied to production systems because of the risk of breaking working applications. There is a need to balance conflicting requirements here.

A vulnerability could conceivably persist for quite a while in a database without being exploited. A patch that breaks an application is known immediately. It is understandable why some DBAs do not patch unless they absolutely have to. As a general rule, we should patch security vulnerabilities. If patches introduce other flaws and cannot be installed without risking even greater harm to the business, than the potential harm due to the vulnerability, then other mitigating controls should be in place to protect the database. DBAs, network managers, and security professionals should work together to formulate the best alternative response to patching.

Software vulnerabilities can also be addressed by removing unnecessary components. Database vendors are including optional functionality—such as support for XML, data mining, text searching, spatial data, and other specialized application areas; the code implementing these components may have vulnerabilities. Thus, unless they are needed, they should not be installed. In addition, depending on your licensing agreements, reducing the number of optional components may also reduce your licensing costs.

Configuration Vulnerabilities

Sometimes we introduce vulnerabilities when we install databases. For example, database systems typically include default accounts, stored procedures, example databases, and so on. Attackers might try to exploit these and therefore should be removed or reconfigured. For example, an attacker may have found a way to elevate privileges by running exploit code in a non-privileged account—what better account for the attacker to try first than a default user account protected with a default password. If that does not work, the next step for an attacker might be to try cracking other accounts.

Weak authentication policies are another configuration vulnerability. During vulnerability assessments, review password policies, including the minimal strength of passwords, the use of generated passwords, and the time between changes. While you are reviewing password policies, also review auditing policies. One must be careful with low-level auditing because an active database can generate large volumes of audit detail. Keep a risk management perspective when deciding what level of auditing to use. Focus on events that could cause the most harm, such as creating a new privileged user or querying large volumes of confidential data.

Finally, ensure the database listener is properly configured and securing controls are in place. The database listener is like the front door of the database: most of what comes and goes to and from the database depends on the database listener at some point. It is an obvious target for attackers.

Application Vulnerabilities

Database applications are technically not part of the database but instead are users of the database. For our purposes, we'll ignore that fact and include application vulnerabilities in database assessments.

The SQL injection attack is probably the most well-known form of database application vulnerability. SQL injection attacks take advantage of developers who trust users to provide properly formatted input that can be directly embedded into a SQL query. Attackers can exploit this fact by crafting input that forces the SQL query to return additional data not intended by the developer. Code reviews, either manual or automated, are required to ensure that user input is scrubbed and properly embedded to prevent a change in the SQL command created by the developer.

In addition, we earlier noted another application vulnerability—access controls and privilege levels can fall out of date because authentication and authorization is not coordinated with centralized identity management systems.

These database vulnerabilities are enough to keep DBAs and network managers up at night, but they are not the only ones.

Vulnerabilities in Data Management Practices

Even if we had an ideally configured and secured database management system, we could still find potential vulnerabilities in the way we manage data. Lack of data classification can result in improper handling of data because we could spend time on non-critical, less-sensitive systems while ignoring systems that are more vulnerable. For example, private data, such as customer credit card data, and confidential data, such as trade secrets, demand more control than publicly available data, such as the addresses of corporate offices. As part of the vulnerability assessment process, ensure a data classification policy is in place and properly enforced with controls such as appropriate authorizations.

Also watch for poor data duplication controls during the vulnerability assessment. For example, developers may copy production data for testing purpose without purging protected information. Another potential vulnerability is insufficient tracking of backup data. Are we sure all backup tapes with private and confidential data are accounted for and no disgruntled employee left with some of those tapes before being hired by a competitor? Similar considerations apply to data maintained at disaster recover sites.

If you do discover weakness such as these, it may indicate a problem with documenting compliance with government and industry regulations. If these problems exist, how is it that the company could demonstrate compliance?

Conducting a Vulnerability Assessment and Prioritizing Mitigations

The following basic checklist outlines the major steps in conducting a database vulnerability assessment:

1. Perform database discovery as described in the first article in this Essential Series.
2. Perform penetration tests on both database management software and operating system (OS) software of the server hosting the database. These are two different types of systems and may require different vulnerability scanners.
3. Conduct a security audit on the database management software and the OS. Audits review information collected about user activities on the database and review policies, roles, and controls in place. Be sure to review privilege abuse and deactivate out-of-date accounts.
4. Review database configurations looking for errors in configuration, default configurations, and unnecessary components.

The findings of this process are then used for the next step: prioritizing mitigations. There is no single order of priorities for all businesses and organizations; however, the following list highlights several factors to consider as well as the demands of business strategy:

- Which vulnerabilities contribute to the risk of data breach, especially for sensitive and confidential data?
- Could any of the vulnerabilities result in violation of government or industry regulations?
- What types of threats could be realized by exploiting the various vulnerabilities and what would it cost to recover from such an incident?
- Do any of the discovered vulnerabilities present significant risk to ongoing, day-to-day operations?

Vulnerability assessments, like database discovery, can provide valuable information for improving database security and compliance. In the next article, we will focus our attention on the next logical step in this process: mitigating risks and monitoring database activity.