White Paper

McAfee
An Intel Company

McAfee Real-Time Database Monitoring, Auditing, and Intrusion Prevention

Table of Contents

Databases store an enterprise's most valuable information assets, but in most organizations today, they are poorly protected. It's good common sense to secure databases as well as or better than other systems in the enterprise. But it's not that simple. In this white paper, we take a look at the shortcomings of the database security tools used by enterprises worldwide and the technology and approach required to provide real-time, comprehensive protection for all types of databases.

Components of Database Security

A wide array of technologies and tools are currently used to provide various aspects of database security. As with other areas of IT security, no single tool can provide an ironclad defense against all threats and abuses. It's best to implement a combination of tools to achieve adequate security. The two main categories of tools in use across enterprises large and small are authentication/access control and native database audit tools.

Authentication and access control

The ability to designate roles, logins, and passwords is the most basic and most widely used level of database security. It establishes privileges for various users and ensures that each user and application have database access to the extent that they need it.

Authentication and access control mechanisms assume, however, that users are generally well-behaved and that their access rights are managed according to policy. Unfortunately, this is often not the case. Granting excessive privileges is commonplace and often results in "privilege creep," where users gain higher levels of privileges over time without having redundant ones revoked. Implementing group usernames and passwords and failing to revoke privileges of employees who no longer need them are also common practices. Furthermore, in many cases, the database lacks an authorization scheme—by default, administrators end up with unlimited access to sensitive data, and it becomes impossible for such privileges to be revoked.

In short, while authentication and access control mechanisms are necessary, they do not suffice even to limit authenticated user access. Additionally, they are vulnerable to exploits like SQL injections that escalate privileges.

"I hack, I ruin, I make piles of money."

—Andrew Auernheimer

(Charged in US Federal Court with stealing and distributing the email addresses of about 120,000 Apple iPad users.*)

Authentication and access control

- *Pros*—Establishes roles and privileges and provides the most basic level of security
- *Cons*—Difficult to enforce properly, access is often granted too liberally, privilege creep is rampant, it's easy to hack, and privileged users are given free reign

* http://www.justice.gov/criminal/cybercrime/auernheimerArrest.pdf

Native database audit tools

Most database management systems (DBMSs) come with built-in features that enable granular auditing of particular database activities. However, in the case of highly transactional environments, or when "selects" (read-only queries, which typically account for the vast majority of database transactions) need to be audited, the performance impact from using these tools can be detrimental. That's why they are used very selectively by most companies.

Native database audit tools

- *Pros*—Provide granular audit trail and forensics of database activity
- *Cons*—Can negatively impact database performance and there's no separation of duties: it's easy to turn off and manipulate, it provides only after-the-fact forensics, and has no prevention capabilities

Furthermore, because native database audit tools are part of the native DBMS toolset, they are usually administered by database administrators (DBAs), violating the segregation of duties mandated by most security and compliance policies. These tools are not a viable solution for monitoring DBAs—or other users who have privileged access rights to the DBMS—because DBAs can turn auditing on and off as they please or manipulate the logs after the fact.

The Network Approach to Database Activity Monitoring

The first generation of dedicated database activity monitoring tools consisted primarily of network-based appliances. These network-based hardware solutions monitor network traffic, looking for SQL statements, analyzing the statements based on policy rules, and creating alerts on any illegitimate access to the database and potential attacks. These appliances monitor only the network—they do not have visibility into local database activity. The database is therefore vulnerable to insiders that either have local access or are savvy enough to bypass the appliances. To provide adequate coverage, the appliance must be deployed at every choke point on the network from which the database is accessed, encircling the database from all sides. For mission-critical databases connected to a multitude of applications (ERP, CRM, business intelligence, billing, and more), covering all access points would require significantly additional expense.

Why are most network monitoring approaches to database security inadequate? Let's take a closer look.

The local access challenge

One obvious problem is the inability to monitor all local access to the database. The typical network monitoring device that captures and analyzes packets from the network will not see local access using IPC or even TCP mechanisms. To overcome this problem, some intrusion prevention system (IPS) vendors have introduced agent-based solutions in addition to their network-based appliances. This approach (installing agents both on the host and in the network infrastructure) removes the only advantage that network appliances have—namely, their relatively non-intrusive nature. Worse still, if there is a local IPS agent, it can monitor TCP traffic only. (Some agents can also monitor IPC communications, but they are even more intrusive and difficult both to install and maintain, since they must be implemented as a kernel module.) Thus, to truly monitor local database activity, it is not enough to capture network traffic, even if you are able to monitor IPC kernel calls.

Let’s take a simple example: you would like to monitor all access to the “customers” table. All network monitoring tools should alert you to the following query:

```
“select * from customers”
```

But what will happen when the next query is run (wherev _cust is a view based on the “customers” table)?

```
“select * from v_cust”?
```

For monitoring tools to actually catch this database access attempt, they have to load and cache all views from the database and understand that the “v_cust” view is selecting from the “customers” table. Other objects, such as synonyms, triggers, and stored program units that could carry out unauthorized or malicious intent are even harder to detect. To understand if a procedure is accessing a specific table, a network monitoring tool must parse the procedure and understand all procedure branches and cases. No network monitoring tool has yet done this, and it’s not feasible for network monitoring manufacturers to develop this capability, as it would require building in a lot of the DBMS’s internal logic.

Pattern matching to catch suspicious activity poses another problem for network monitoring devices. A monitoring tool can be configured to catch “grant dba” commands—for example, if a hacker tries to mount an SQL injection attack using a known Oracle vulnerability like the one below:

```
declare
l_cr number;
begin
l_cr := dbms_sql.open_cursor;
dbms_sql.parse(l_cr,’declare pragma autonomous_transaction; begin execute immediate ‘’grant dba to
public’’;end;’, 0);
sys.lt.findricset(‘.’’||dbms_sql.execute(‘||l_cr||’)||’’’,’x’);
end;
```

Most monitoring tools catch this procedure and issue an alert because they match the pattern of “grant dba” and the existence of a vulnerable package. However, if the hacker is smart, he will try to evade detection by performing the same attack differently:

```
DECLARE
l_stmt VARCHAR2(32000);
BEGIN
l_stmt := utl_encode.text_decode(‘
CmRlY2xhcmUKICAgIGxfY3IgbnVtYmVyOwpiZWdpbgogICAgbF9jciA6PSBkYm1z
X3NxbC5vcGVuX2N1cnNvcjsKICAgIGRibXNfc3FsLnBhcnNlKGxfY3IsJ2RlY2xh
cmUgcHJhZ21hIGF1dG9ub21vdXNfdHJhbnNhY3Rpb247IGJlZ2luIGV4ZWN1dGUg
aW1tZWRpYXRlICcnZ3JhbnQgZGJhIHRvIHB1YmxpYycnO2VuZDtlbmQ7Jywg
MCk7CiAgICBzeXMubHQuZmluZHJpY3NldCgnLicnfHxkYm1zX3NxbC5leGVjdXRl
KCd8fGxfY3J8fCcpfHwnJycsJ3gnKTsKZW5kOw==’, ‘WE8ISO8859P1’, utl_encode.base64);
   EXECUTE IMMEDIATE l_stmt;
EXCEPTION
   WHEN OTHERS THEN NULL;
END;
```

Notice that there is no longer a “grant dba” string in the text; network-based systems will be blind to what is really transpiring. The hacker even disguises the call to the vulnerable function.

The encryption challenge

Encryption is also troublesome for network monitoring systems. Malicious database traffic can be encrypted using vendor-supplied tools or custom tools like SSH tunneling. For network monitoring tools to capture this type of traffic, an enterprise must compromise its private keys and share them with the monitoring application. In addition, database traffic can be encrypted, and decrypting it in real time is simply not possible even if the encryption algorithm is known, which is unlikely. (Oracle, for instance, does not make its encryption algorithm public.)

If a DBA or external hacker had created a function like the following one, it would have raised the suspicion of the monitoring tools:

```
CREATE OR REPLACE FUNCTION get_dba
RETURN VARCHAR2
AUTHID CURRENT_USER
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
  RETURN 'Hacked';
END get_dba;
```

However, by simply creating the function using the database's built-in wrap utility, the function would not have attracted any attention:

```
CREATE OR REPLACE FUNCTION get_dba wrapped
a000000
b2
abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd
8
a6db
7EiybMnZ7oeJndiapoeSr+FlvzQwg2LwLcsVfHSikx5kpaeQDbcTdSGEdl1X42LFoOBwQ7Xp
RrTcu0G50S40Y2bOeylQqn4Ofi5ElBo/bAAdKrpeZ5rDk9jEl54mFfVcGFi4d+ny0TufXvHy
nQ2Ib0qhcAba+MlfPhfL9GDAUhFAOKigrD0fgnhq0p0yHjPPLPjKVvvvuiwGz5LhRNVWjA==
```

McAfee Database Activity Monitoring: Real-Time Database Monitoring and Intrusion Prevention

If you look at each of the problems described in the last section, coupled with the fact that all monitoring solutions now offer an agent in addition to the network device, it's obvious that enterprises need a better way to monitor database activity—one that uses a non-standard method of accessing the database. To be able to catch infiltrations like those described—and countless variations and permutations—an effective solution must be able to interact intimately with the database. This is exactly what McAfee® Database Activity Monitoring does.

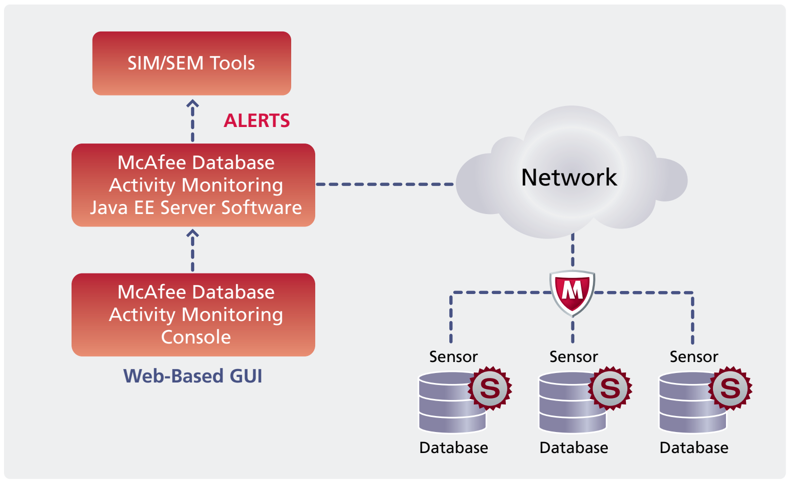
SIM/SEM Tools
ALERTS
McAfee Database Activity Monitoring Java EE Server Software
McAfee Database Activity Monitoring Console
Web-Based GUI
Network
Sensor
Sensor
Sensor
Database
Database
Database

Figure 1. McAfee Database Activity Monitoring architecture.

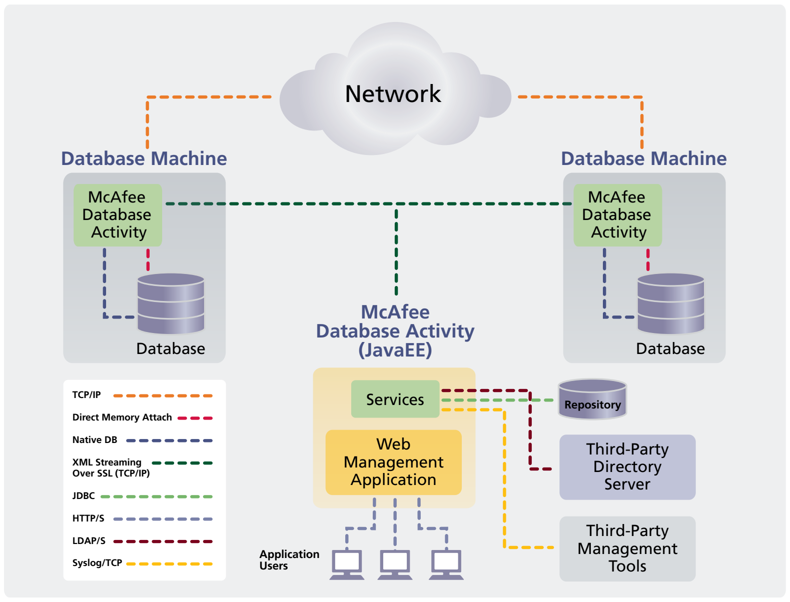

Figure 2. Detailed architecture.

McAfee Database Activity Monitoring consists of a management console and small footprint sensor: a software agent that is installed on the database host server and monitors all database-related activity. The sensor is a stand-alone process written in C++ that runs on the database host machine. It is installed using standard platform tools (RPM, PKG, DEPOT, BFF, EXE) in a separate operating system user account on the system. The sensor automatically identifies all instances on the machine and can monitor multiple instances, even different database types, on the same host. It is non-intrusive and consumes only a fraction of CPU resources (less than five percent of a single CPU core, even on machines with multiple CPUs).

When running, the McAfee Database Activity Monitoring sensor attaches itself to the database's instance memory in the area of the SQL cache. Using read-only mechanisms and application programming interfaces (APIs), it begins a polling loop that monitors the database by continuously sampling its memory. For every sample cycle, the sensor analyzes currently running and previous statements in the database instance and, using a predefined policy received from the server, determines which statements should trigger alerts or be blocked. Statements violating the predefined policy are sent to the management console in real time as alerts. The sensor can also be configured to terminate database sessions on specific violations, without introducing any risk to data integrity, and to quarantine users.

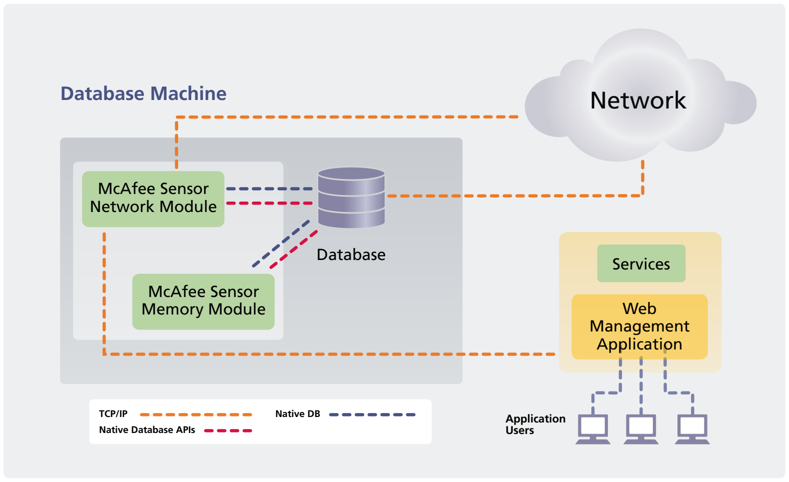

Figure 3. McAfee Database Activity Monitoring sensor architecture for Oracle.

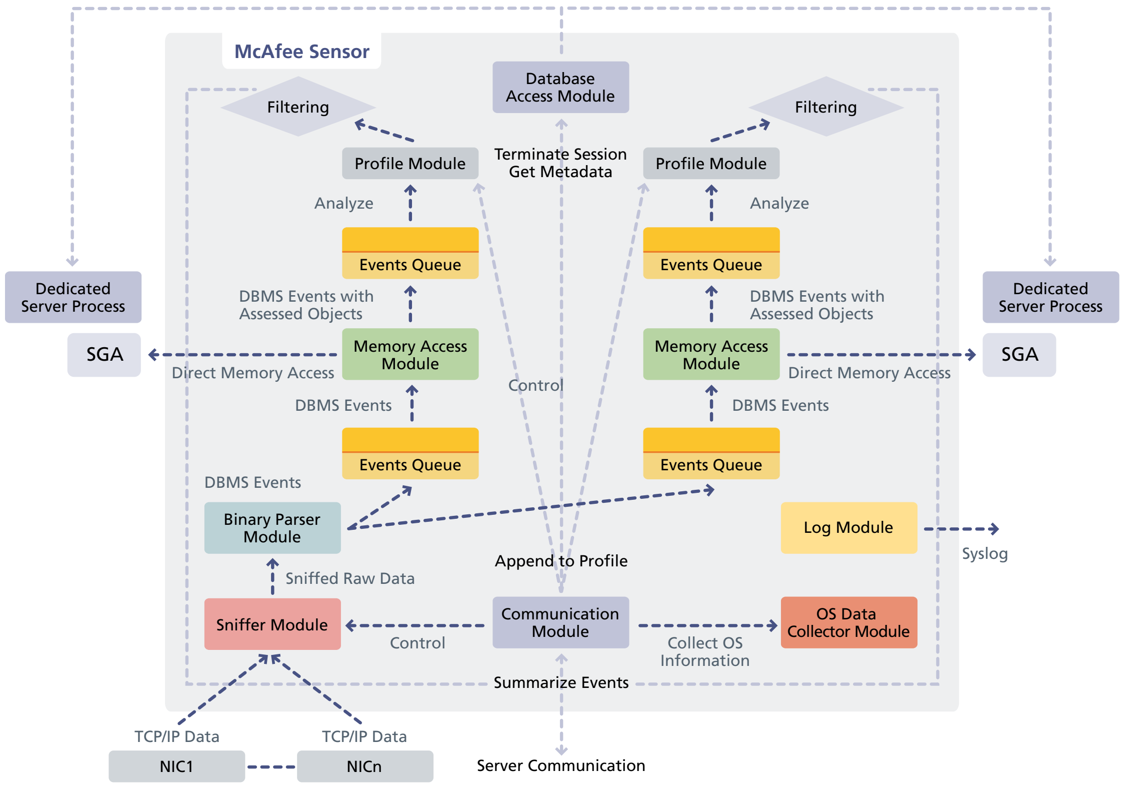

Figure 4. McAfee Database Activity Monitoring sensor data and control paths.

Policy rules for McAfee Database Activity Monitoring consist of keywords: command-type, SQL statement, application-name, time-of-day, object, user, and more, combined with operators such as equal, contains, matches (for regular expressions), and others. Logical operators such as and, or, and brackets can be used as well. The rules also contain the actions taken when the conditions of a rule are met—for example: send an alert to a SIEM/SEM system via SNMP, Syslog, or XML API; send an email; terminate a user session to prevent malicious activity; or even quarantine users. The system also comes with predefined policies called vPatch rules, short for "virtual patching," that prevent attacks that exploit known database vulnerabilities, as well as generic rules based on context and patterns that prevent zero-day exploits.

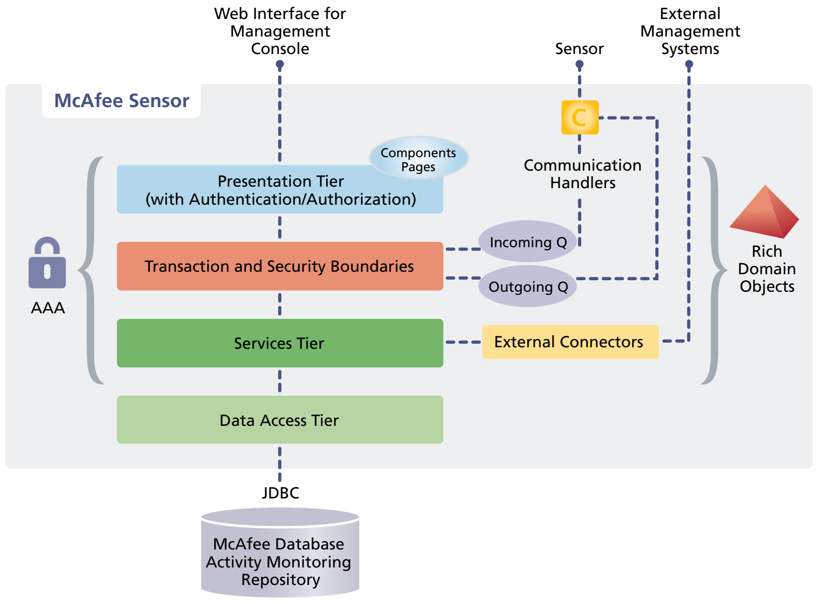
Web Interface for Management Console
Sensor
External Management Systems
McAfee Sensor
Components Pages
Presentation Tier (with Authentication/Authorization)
Communication Handlers
Incoming Q
Transaction and Security Boundaries
Outgoing Q
AAA
Rich Domain Objects
Services Tier
External Connectors
Data Access Tier
JDBC
McAfee Database Activity Monitoring Repository

Figure 5. McAfee Database Activity Monitoring server architecture.

A single McAfee Database Activity Monitoring server can manage and communicate with numerous sensors on different databases. An enterprise installation can easily scale to encompass hundreds of databases. The server also easily integrates with your IT infrastructure to facilitate central IT management and security event management.

Since the McAfee Database Activity Monitoring sensor resides on the database machine, it cannot be bypassed; its self-defense mechanisms will detect any tampering attempts immediately and will trigger an alert. The structure of the McAfee system also enables separation of duties, a key compliance requirement. The McAfee Database Activity Monitoring administrator, the person defining policy rules, and the person receiving alerts can all be different people in different departments within the organization—for example, IT manager, DBA manager, and CISO, respectively.

To keep policies as up to date as possible, McAfee employs a "red team," a group of penetration testers who constantly test for new database vulnerabilities. As soon as new vulnerabilities are discovered, the red team creates rules to protect against their exploitation. These virtual patches immediately protect the database, without the need for system upgrade or downtime, keeping the database from being exposed until DBMS updates are issued by the database vendor.

Unique advantages

McAfee Database Activity Monitoring offers a truly unique approach to database security:

- The only database monitoring solution that monitors all database activities and provides protection against insiders with privileged access
- Granular monitoring of database transactions, queries, objects, and stored procedures, with real-time alerts and breach prevention
- Flexible rules that allow enforcement of corporate security policy with minimal "false positive" alerts
- Virtual patches for newly discovered database vulnerabilities, providing immediate protection with no DBMS downtime
- Flexible audit and reporting capabilities
- Easy-to-deploy and scalable software solution
- Separation of duties—a key compliance requirement

For more information, visit www.mcafee.com/dbactivitymonitoring, or contact your local McAfee representative or reseller near you.

About McAfee Database Security

McAfee database security solutions offers real-time reliable protection for business-critical databases from both external and intra-database threats, requiring no architecture changes, costly hardware, or database downtime. Organizations can gain complete visibility into their overall database landscape and corresponding security posture, fully align their security policy administration practices, and efficiently maintain regulatory compliance.

About McAfee