Introduction to Database Log Management

Anton Chuvakin, Ph.D.
Chief Logging Evangelist
LogLogic, Inc

Updated: 08/08/2007

NOTE: Originally published in CSI Alert Newsletter (February and April 2007)

Part I

Database security have been capturing more and more attention in recent years, even though most of the security issues surrounding databases existed since the first day commercial database systems were introduced in the market in the 1980s.

Nowadays, database security is often seen as containing the following principal components:

- access control to database software, structures and data
- database configuration hardening
- database data encryption
- database vulnerability scanning

It is interesting to see that logging and auditing underline all of the above domains of database security. Indeed, the only way to verify what access control decisions are being made and who views what data from the RDBMS is to look at the authentication logs. Database configuration hardening includes enabling and increasing the auditing levels. Similarly, data encryption might be verified by log and configuration review. And, vulnerability exploitation usually leaves traces in logs despite what some say (the challenge is more often with understanding what the log said and not with having the logs) In recent years, insider attacks gathered more attention than periodic outbreaks of malware; and database logging happens to be in the forefront of this fight against insider attacks. Database systems are usually deployed deep inside the company network and thus insiders are usually has the easiest opportunity to attack and compromise them, and then steal (or “extrude” as some would say) the data.

In addition to security, database logging (as well as log analysis) needs to be performed to allow IT auditors to do their jobs and enable regulatory compliance validation. On a more general level, IT governance and “best practice” frameworks such as COBIT or ITIL also steer IT users towards database logging.

Different database software offers a dizzying array of logging options. For example, Oracle, the leading database software, has mandatory, standard and fine-grained audit. Specifically, to enable mandatory auditing (off by default) one needs to set the *audit_sys_operations* parameter to true for the appropriate database instance. Enabling standard audit is done by setting the *audit_trail* parameter for the instance and then enabling the relevant audit options.

Similarly, MS SQL Server would always log server access and errors to a Windows event log, but logging – or "tracing" – other queries requires running a separate tool that creates trace files.

And, finally, the open source MySQL has several log types – error log, query log, binary log of statements that change data and slow query logs. While error logs are always enabled, one needs to manually enable other types of logging. For example, one enables query logging by starting mysql as "*mysql –log*".

So what is in common among the database logging? The above databases would log (rather, can be configured to log as most would log very little by default) most if not all of the following events:

- user logins and logouts
- database system starts, stops and restarts
- various system failures and errors
- user privilege changes
- database structure (metadata) changes
- most other DBA actions
- select or all database data access (if configured to be so)

The above logged events provide rich sources of information for the mentioned security, compliance and IT operational needs. A better way of saying it would be "would have provided if they were easy to analyze." Many factors, such as log availability, log format complexity and log volume, hinder the log analysis efforts and most apply to database logging.

What are the possibilities for database log formats? Given that databases are built to store structured data, one would assume that logs would always be stored in a database itself. That assumption will in fact be wrong. Most databases use a mix of logging to itself and logging to flat files, sometimes binary files. For example, in case of Oracle, database administrator (DBA) actions are logged into a separate file in order to prevent said DBAs from modifying the logs of his own actions (typically, a DBA will have unfettered access to the RDBMS, but not 'root' or Administrator on the underlying system). Here is an example of such log:

Tue Sep 19 15:17:39 2006
ACTION : 'SHUTDOWN'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle
CLIENT TERMINAL: pts/4
STATUS: 0

At the same time, Oracle database audit tables contain other audit information such as time, user name, SQL command, status and other data access auditing parameters.

Microsoft SQL Server database uses Windows event log facility to log its operational messages – such as starts, stops, backup status, etc - as well as server access events - user logins and logouts. However, file access audit logs are done differently by using so-called “trace files” – binary files that contain the information on all database data accesses. Here is an example messages produced by MS SQL Server:

- “There is insufficient system memory to run this query.”
- “The master database failed to restore. Use the rebuildm utility to rebuild the master database. Shutting down SQL Server.”
- “Login failed. The maximum simultaneous user count of 1 licenses for this 'Standard Edition' server has been exceeded. Additional license should be obtained and installed or you should upgrade to a full version.”

It is important to realize that default logging options on most databases exclude logging of access to data, such as the SELECT statements as well as changes to data tables such as UPDATE or DELETE. Even more drastic actions such as table DROPs (used to detete tables) and ALTERs (used to change table structure) are not recorded, unless specifically configured by the DBA. To enable data-level logging one needs to set special configuration options and sometimes even to restart database software.

Unlike other situations where logging has minimum impact on system performance, database audit logging does slow down the database, sometimes significantly. After all, high-performance databases are meant to provide thousands of data transactions per second and logging all of these presents a challenge to system IO and well as CPU and disk storage resources. That is why logging has somewhat of a bad rap with DBAs. The fact that logs can and should be used to track DBA actions for possible violations does not work to improve that perception at all ☺ However, there are more and more reasons to enable logging nowadays and thus such performance considerations increasingly become secondary.

If one tries to log even a subset of all data access and data changes, he should be prepared to drown in an ocean of audit log data. Extracting useful actionable information from such onslaught of data presents a challenge.

Apart from performance and data volume, other challenges with database logs include verbosity and obscurity of log formats. Multi-line formats where a single message spans multiple lines also hamper automated analysis of logs.

Given the situation above, what should we do to make sense of database logs? While detailed discussion of database log analysis goes outside the scope of this introductory paper, we will try to give a few useful hints.

First, as all other logs, database logs are indispensable for incident response. Thus, even collection of logs with no analysis whatsoever is going to help you during the crunch time of a data theft, server crash or even a surprise visit by your friendly auditor. If you are to take this one step, collecting logs from multiple serves at one central location (such as your log management solution) will make incident analysis much simpler and will prevent loss of log data due to routine log rotation or other reasons. Recent data theft incidents are making this motivation for database log analysis more powerful every day (and every new breach!)

Beyond just keeping logs for the “rainy” day one should institute a period review of a DBA activity logs. Such logs are usually less voluminous than data access audit, but provide critical information on DBA actions important for keeping tabs on people entrusted with your “crown jewels” such as customer lists or product inventory information. In some cases, even a manual review or a set of scripts will work, but for better efficiency, a log management tool will do the trick much better.

Further, if you need to log access to some data, which increasingly happens due to various regulations and “best practices” frameworks, manual log review will break down pretty quickly due to log volume (you might be looking at gigabytes of data per day or more) Thus, you will be embarking on a trail towards automating your databases log collection, analysis and archival by deploying log management tools.

To conclude, database logging represents a new world for many security practitioners, but the one which they might have no choice but conquer. Enabling logging is a good start, from where one can progress towards more in-depth log analysis.

Part II

In this part of the paper, we will focus on what happens after you decide to enable all the logging we suggested in the previous part.

While currently firewalls still hold the dubious record of spewing the highest volume of logs, database might rival such log volume once more people configure them to enable logging motivated by security, compliance or broader operational reasons and not just narrow "keeping the database running."

We mentioned that by default most commercial databases record surprisingly few events of interest. Thus, a manual log review may be suitable under such circumstances, given that most of important events don't ever show up in logs.

Automation in the form of log analysis and log management tools becomes essential if any kind of database security "best practices" are being followed and logging is enabled.

So, what might such tools provide? Simpler log analysis tools will allow users to review at logs on a specific database server only and likely require such tool to be running on that same server. Consequently, such tools will only work for a specific database type such as Oracle-only or DB2-only. They also usually do not provide any real-time analysis in the form or alerting or triggering automated actions. Still, they can provide some insight s from logs, which are clearly superior to going through raw logs line by line.

Database vendors usually provide such tools. Examples include IBM *DB2 Log Analysis Tool* (that can be used to generate simple database log reports for a single database server), *MS SQL Profiler* (that can be used to create and review audit trace files from a SQL Server) and *Oracle LogMiner* (that can analyze Oracle redo logs)

More advanced log management tools will work across multiple database servers and even across database types. Many would also allow analysis of database logs in combination with other logs such as server, security device or firewall. Analysis of this type allows put the database data in the context or other organization log data and to correlate database activity with other things that happen at the organization at the same time.

Such tools automate not only analysis of log data, but the whole lifecycle of log management:

- Collecting the log where it is being generated via an agent or remotely
- Securely transferring the data to the central server for analysis and storage
- Issuing real-time alerts to database administrators if needed
- Providing reports and analytics based on log data
- Securely storing the logs as long as prescribed by the retention policy and then, just as safely, destroying them

Let's look at each of the stage of the above process and see how it applies to databases.

First, how do we collect database log data for analysis? As we learned in the previous part, database might be logging to its own tables or to dedicated files, either text or binary and proprietary. One obvious solution is to use a dedicated piece of software - an agent (however hateful the word can be in some circles!) – running on a database server which allows for efficient log collection and secure data transfer. However, many DBAs would strongly object about running something extra on "their" servers, and, given typically high requirements for database server uptime, such concerns are not entirely unfounded. Thus, a remote or "agentless" approach needs to be used if possible. If a database logs are located into a database tables, one can connect to said server via ODBC or JDBC and grab all the needed data by executing a SELECT command. This allows for granular audit collection which will not endanger database operation. At the same time, many databases now natively support SSL for JDBC/ODBC transfers and thus our data connection will be secured against eavesdropping. Once the log data has been collected by the log management tool, it can be purged from the original database to save space. In this case, a log management tool will have a full set of logs from all the databases.

In case of a file-based logging, such as in the case of Oracle DBA logs that cannot be directed to a database for storage, a secure SFTP or FTPS connection may be used to grab the logs, also in a secure and authenticated fashion. Needless to say, such log grabs are logged as well.

Second, while logs are being moved to a log management tool, time-sensitive alerting needs to take place. For example, if we observe that a somebody is downloading an entire table or changing a database scheme while being logged from a remote connection, it might be worth an alert. More sophisticated algorithms, such as anomaly detection or correlation, may be used to drive alerting as well.

Third, more in-depth analysis of collected log data is performed via reports and searches. Why review the database log reports? The reasons we show below are driven by operational, security and regulatory or compliance concerns. Specifically,

- **Change management**: modern RDBMS is a complicated piece of software with plenty of configuration files and other things to change. And being aware of all the changes constitutes one of the control objectives for IT governance as well as essential for protecting the environment.
- **Authentication, Authorization and Access:** logging access control decision such as login failures and successes as well as access to data and various database objects is of interest for auditors as well as important for security, such as insider privilege abuse. While logging all access to data is less common, it is one of the emerging trends in logging. At the same time, logging clearly unusual access, such as multiple massive bulk data reads performed from a regular client workstation, definitely need to be "on the radar."
- **Threat Detection**: while both unauthorized changes and access control decisions are essential for security, database logs may be used to discover and analyze direct exploitation attempts as well, at least in some cases. Things like accessing the database system at an unusual time (off-hours) especially with a DBA-level access privileges will likely be of interest to as security team.
- **Performance**: DBAs are tasked with monitoring database performance and logs provide one of the avenues for doing so. This is especially important to those orgs with strict service level agreements (SLA) for database performance
- **Business Continuity**: knowing of database software starts and stops is essential since business depend on databases for their revenue streams and this a downed database directly leads to losing money. Another twist on continuity is tracking backups: are they done? Are they usable? Who has access to them?

Thus, looking at the database log reports that cover the above categories would be of value for multiple stakeholders within the organization and even beyond (in case of an external audit)

Finally, retaining log data is now directly mandated by some of the regulations and industry guidelines such as PCI (which covers database logs in particular since databases are often involved in processing credit card data). Log management tools will help you automate such policy driven retention and make sure that data from each database or a group of database servers is retained as long as needed and then destroyed. Online retention period will commonly spam 3-12 month period with longer retention on tape or other dedicated storage tools.

Making a choice between simple analysis tools and a log management deployment depends upon the usual factors such as available budget, IT resources, size of an organization as well as regulatory mandates. Obviously, deploying any solution enterprise-wide is a more involved process than simply installing a tool on an individual server, but benefits of such broad deployment will go far beyond solving a single tactical problem. Thus, author's recommendation will be to combine database log management with other similar projects (such as firewall or Unix server syslog management) and use a single platform for all of them, but grow such deployment in phases, rather than try to cover all logs on day one.

One final issue that needs to be addresses is performance. Given that database administrators are very sensitive about everything that may slow down their databases, we need to provide a few pointers.

If agent is deployed on a database server for log collection, it may indeed slow down the system, either during its normal operation or if a bug happens to affect the agent software (this can't happen to you, can it? ☺). For example, a memory leak in the agent can cause memory exhaustion on the server which will adversely affect the database performance.

Remote log collection is much less likely to affect the database performance, but the initial retrieval of a large bulk of log records from a database has a potential of slowing down other operations, at least for IO reasons if not for others. To avoid this problem, one can schedule log collection to occur at off ours, when all other database service operations, such as backups, happen. At the same time, direct log grab from a database may be more bandwidth-intensive than the agent communication, which then starts to win out if logs need to be grabbed from remote databases. Still, given that database are commonly deployed in a datacenter, availability of high-bandwidth network pipes is almost assured.

Overall, given that to satisfy current operational, security and compliance requirements databases need to be configured with detailed logging which will result in a flood of audit log data, using log management tools for log automation is absolutely essential. While database-vendor-specific tools can help a bit, a comprehensive log management solution presents a better choice in most circumstances.

We would also like to note a couple of future trends that affect database logging and log management. One is an overall increase in logging across IT solutions. It would not be surprising if in a few years many organization will have to log all access to data in their critical databases and their other data storage systems. The second trend that will help allay the consequences of the first one is an increasing level of intelligence in log management tools. Thus, despite the dramatic increase of log data volume, it is likely that workloads of the IT teams

will not grow at the same pace since many of the tasks, even deeper analysis tasks, will be automated.

Dr Anton Chuvakin (http://www.chuvakin.org) is a recognized security expert and book author. In his current role as a Chief Logging Evangelist with LogLogic, a log management and intelligence company, he is involved with projecting company's vision of log management, driving the product roadmap, conducting research as well as assisting key customers with their LogLogic implementations.

A frequent conference speaker, he also represents the company at various security meetings and standards organizations. He is an author of a book "Security Warrior" and a contributor to "Know Your Enemy II", "Information Security Management Handbook", "Hacker's Challenge 3" and "PCI Compliance." In his spare time he maintains his security portal http://www.info-secure.org and several blogs such as http://chuvakin.blogspot.com.