Understanding and Selecting a Database Security Platform

An update to *Understanding and Selecting a Database Activity Monitoring Solution.*

Version 2.0

Released: May 15, 2012

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog but has been enhanced and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Licensed by Application Security Inc.

About Application Security, Inc.

AppSecInc is a pioneer and leading provider of database security solutions for enterprise of all sizes. By providing easy to deploy and manage, highly scalable software-only solutions – AppDetectivePro for auditors and IT advisors, and DbProtect for the enterprise – AppSecInc helps customers achieve unprecedented levels of data security, while reducing overall risk and helping to ensure continuous regulatory and industry compliance. Used by more than 1,300 active commercial and government customers worldwide, our proven and award-winning enterprise solutions are backed by the world's most comprehensive database security knowledgebase from the company's renowned team of threat researchers, TeamSHATTER.

For more information, please visit: www.appsecinc.com and follow us on Twitter: www.twitter.com/appsecinc | www.twitter.com/teamshatter.

Licensed by GreenSQL

GreenSQL delivers Unified Database Security Solutions for the small and medium size businesses (SMB). The company is committed to protecting information by making database security affordable and easy to manage for every company. With an all-in-one approach to database security, the GreenSQL software-based platform offers Database Security, Caching, Auditing and Dynamic Data Masking in a single package.

Visit GreenSQL at www.greensql.com

Licensed by Imperva

Imperva is a pioneer and leader of a new category of data security solutions for high-value business data in the data center. With more than 1,800 end-user customers and thousands of organizations protected through cloud-based deployments, Imperva's customers include leading enterprises, government organizations, and managed service providers who rely on Imperva to prevent sensitive data theft from hackers and insiders. The award-winning Imperva SecureSphere identifies and secures high-value data across file systems, web applications and databases. For more information, visit www.imperva.com, follow us on Twitter or visit our blog.

Licensed by McAfee

McAfee, Inc., headquartered in Santa Clara, California, is the world's largest dedicated security technology company. McAfee is relentlessly committed to tackling the world's toughest security challenges. The company delivers proactive and proven solutions and services that help secure systems and networks around the world, allowing users to safely connect to the Internet, browse and shop the web more securely. Backed by an award-winning research team, McAfee creates innovative products that empower home users, businesses, the public sector and service providers by enabling them to prove compliance with regulations, protect data, prevent disruptions, identify vulnerabilities, and continuously monitor and improve their security.

http://www.mcafee.com

Copyright

Table of Contents

Introduction

Database Activity Monitoring (DAM) has progressed over the last 10 years from obscure tool, to an adolescent technology for specific database security challenges, to a mature enterprise platform offering significant security and compliance benefits. The threats may change -- networks to hosts, and from hosts to applications -- but the target remains the same: Information. The importance of securing databases, the most common repository for sensitive data, has grown accordingly. DAM solutions provide a powerful, immediate, non-intrusive method for security and compliance, and a long-term platform for comprehensive protection of databases and applications. Anyone doubting about the importance of this technology to data, database and application security should consider that this market has steadily grown in size since 2005, and 7 of the startup firms have been acquired by much larger security and database vendors. DAM is thoroughly entrenched as an enterprise security platform.

Five years ago when Rich authored the white paper '[Understanding and Selecting a Database Activity Monitoring Solution]', he made the following statement:

"Although I call this product category Database Activity Monitoring, I don't believe that name sufficiently describes where the market is headed."

Despite that observation, we remain somewhat surprised by the current state of the market. When we started this research project we believed that it was time to update that paper given that a lot had changed in 5 years. While the core of that paper remains an excellent primer on the subject of DAM, it is missing some significant trends. We thought that refreshing that paper was a simple matter of updating the use cases and listing some of the new capabilities, but the research process uncovered a lot more. One significant discovery was customer use cases were splintering, not converging. We also found that the increased scope of monitoring coverage and security benefits meant an evolutionary step forward, one which the term 'DAM' no longer adequately described. There's a lot more going on here which we -- along with the rest of the industry -- struggle to describe. What we feel makes the most sense, and the best term we have see to date, is Database Security Platform.

Being sensitive to the fact that analyst firms and vendors appear to randomly rename existing markets to differentiate themselves, we did not undertake this lightly. We -- the authors of this research -- have been active participants in the database security market for over a decade and involved with the evolution of Database Activity Monitoring. Rich, during his tenure with Gartner, was the one who coined the term "Database Activity Monitoring" in the first place. And to be honest, we'd prefer not to change, but Mr. Market says it's time to move forward.

Here's the new definition and why we think this is important:

Defining DSP

Our old definition for Database Activity Monitoring is modified to the following:

***Database Security Platforms**, at a minimum, assess database security, capture and record all database activity in real time or near real time (including administrator activity); across multiple database types and platforms; and alert and block on policy violations.*

This distinguishes *Database Security Platforms* from *Database Activity Monitoring* in four key ways:

1. Database security platforms support both relational and non-relational databases.
2. All Database Security Platforms include security assessment capabilities.
3. Database Security Platforms **must** have blocking capabilities, although they aren't always used.
4. Database Security Platforms often include additional *protection* features, such as masking or application security, which aren't necessarily included in Database Activity Monitors.

Key DSP Features

- Vulnerability, user, and configuration **assessment**.
- Monitor **all** database activity.
- Can **block** attacks.

We are building a new definition due to the dramatic changes in the market. Almost no tools are limited to merely activity monitoring any more, and we see an incredible array of (different) major features being added to these products. They are truly becoming a platform for multiple database security functions, just as antivirus morphed into Endpoint Protection Platforms by adding everything from whitelisting to intrusion prevention and data loss prevention.

Here is some additional detail:

1. The ability to remotely audit all user permissions and configuration settings. Connecting to a remote database with user level credentials, scanning the configuration settings, then comparing captured data against an established baseline. This includes all external initialization files as well as all internal configuration settings, and may include additional vulnerability tests.
2. The ability to independently monitor and audit all database activity including administrator activity, transactions, and data (`SELECT`) requests. For relational platforms this includes DML, DDL, DCL, and sometimes TCL activity. For non-relational systems this includes ownership, indexing, permissions and content changes. In all cases read access is recorded, along with the meta-data associated with the action (user identity, time, source IP, application, etc).
3. The ability to store this activity securely outside the database.
4. The ability to aggregate and correlate activity from multiple, heterogeneous Database Management Systems (DBMS). These tools work with multiple relational (**e.g.**, Oracle, Microsoft, and IBM) and quasi-relational (ISAM, Terradata, and Document management) platforms.
5. The ability to enforce separation of duties on database administrators. Auditing activity must include monitoring of DBA activity, and prevent database administrators from tampering with logs and activity records -- or at least make it nearly impossible.
6. The ability to protect data and databases -- both alerting on policy violations and taking preventative measure to prevent database attacks. Tools don't just record activity -- they provide real-time monitoring, analysis, and even rule-based response. For example, you can create a rule that masks query results when a remote `SELECT` command on a credit card column returns more than one row.

7. The ability to collect activity and data from multiple sources. DSP collects events from the network, OS layer, internal database structures, memory scanning, and native audit layer support. Users can tailor deployments to their performance and compliance requirements, and collect data from sources best for their requirements. DAM tools have traditionally offered event aggregation but DSP requires correlation capabilities as well.

DSP is, in essence, a superset of DAM applied to a broader range of database types and platforms.

Platform Highlights

- **Databases**: It's no longer only about big relational platforms with highly structured data -- but now also non-relational platforms. Unstructured data repositories, document management systems, quasi-relational storage structures, and tagged-index files are being covered. The number of query languages being analyzed continues to grow.
- **Assessment**: "Database Vulnerability Assessment" is offered by nearly every Database Activity Monitoring vendor, but it is seldom sold separately. These assessment scans are similar to general platform assessment scanners but focus on databases -- leveraging database credentials to scan internal structures and metadata. The tools have evolved to scan not only for known vulnerabilities and security best practices, but to include a full scan of user accounts and permissions. Assessment is the most basic preventative security measure and a core database protection feature.
- **Blocking**: Every database security platform provider can alert on suspicious activity, and the majority can block suspect activity. Blocking is a common customer requirement -- it is only applied to a very small fraction of databases, but has nonetheless become a must-have feature (at least in the selection process, it often isn’t actually deployed). Blocking requires the agent or security platform intercept the incoming requests before they execute.
- **Protection**: Over and above blocking, we see traditional monitoring products evolving protection capabilities focused more on data and less on database *containers*. While Web Application Firewalls to protect from SQL injection attacks have been bundled with DAM for some time, we now also see several types of query result filtering.

One of the most interesting aspects of this evolution is how few architectural changes are needed to provide these new capabilities. DSP still *looks* a lot like DAM, but *functions* quite differently. We will get into architecture later in this report.

Database Security Platform Origins

Those of you familiar with DAM already know that over the last four years DAM solutions have been bundled with assessment and auditing capabilities. Over the last two years we have seen near universal inclusion of discovery and rights management capabilities. DAM is the centerpiece of a database security strategy, but as a technology it is just one of a growing number of important database security tools. Let's spend a moment looking at the key components, how we got here, and where the technology and market are headed. We feel this will fully illustrate the need for the name change.

The situation is a bit complicated, so we include a diagram that maps out the evolution. Database Activity Monitoring originated from leveraging core database auditing features, but quickly evolved to include supporting event collection capabilities:

- *Database Auditing* using native audit capabilities.
- Database Activity Monitoring *using network sniffing* to capture activity.
- Database Activity Monitoring with *server agents* to capture activity.

So you either used native auditing, a network sniffer, or a local agent to track database activity. Native auditing had significant limitations -- particularly performance -- so we considered the DAM market distinct from native capabilities.

Due to customer needs, most products quickly combined network monitoring and agents into single products -- perhaps with additional collection capabilities, such as memory scanning. The majority of deployments were to satisfy compliance or audit requirements, followed by security.

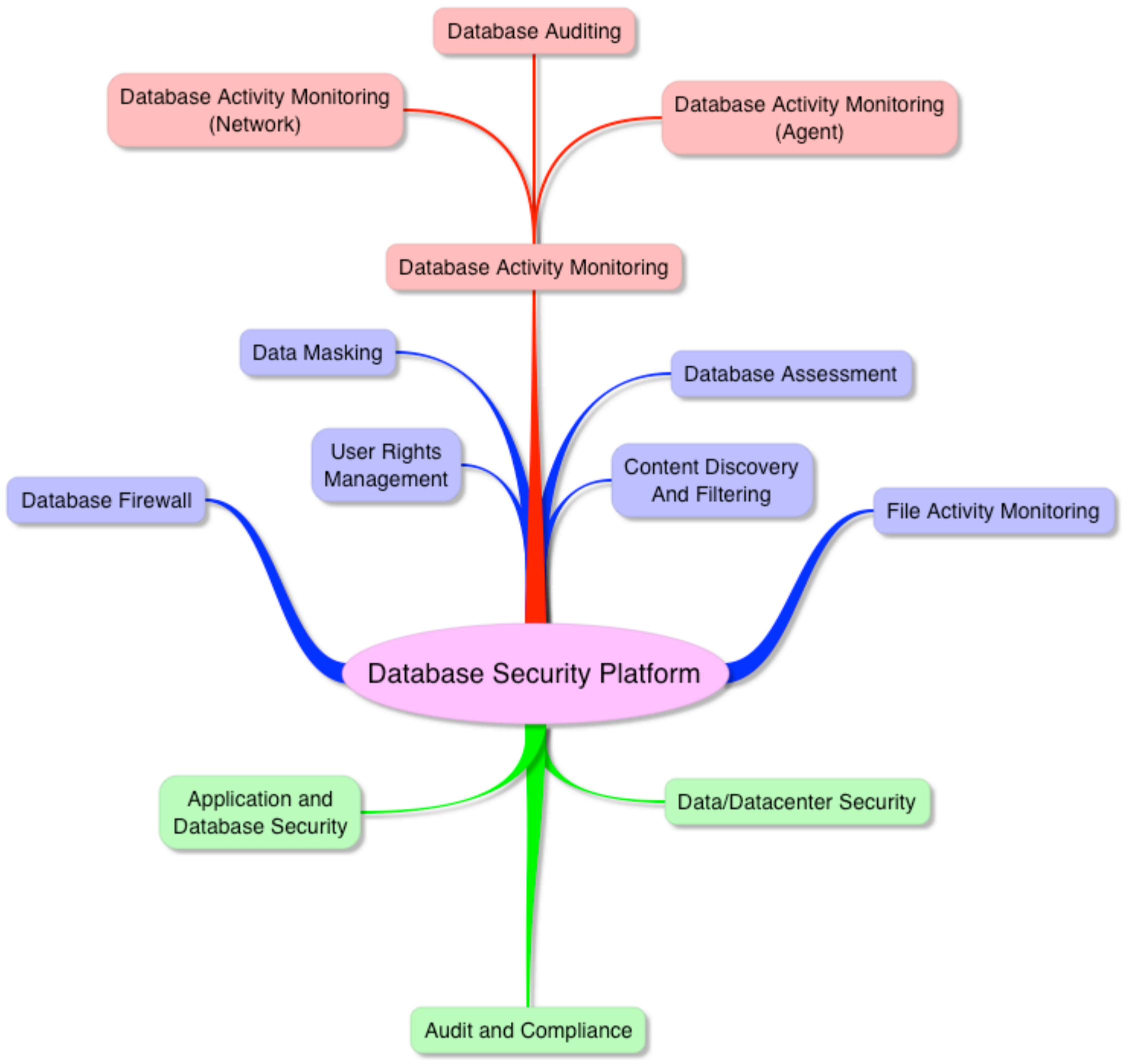
There were also a range of distinct database security tools, generally sold standalone:

- *Data Masking* to generate test data from protection data, and to protect sensitive information while retaining important data size and structural characteristics.
- *Database Assessment* (sometimes called *Database Vulnerability Assessment*) to assess database configurations for security vulnerabilities and general configuration policy compliance.
- *User Rights Management* to evaluate user and group entitlements, identify conflicts and policy violations, and otherwise help manage user rights.

Other technologies have started appearing as additional features in *some* DAM products:

- *Content Discovery and Filtering* to identify sensitive data within databases and even filter query results.
- *Database Firewalls* which are essentially DAM products placed inline and set to filter attack traffic, not merely monitor activity.
- A subset of *File Activity Monitoring* to monitor non-relational database files, configuration files on the server, or even generic files (although we consider that a separate market).

The following graph shows where we are today:


Database Auditing
Database Activity Monitoring (Network)
Database Activity Monitoring (Agent)
Database Activity Monitoring
Data Masking
Database Assessment
User Rights Management
Content Discovery And Filtering
Database Firewall
File Activity Monitoring
Database Security Platform
Application and Database Security
Data/Datacenter Security
Audit and Compliance

As the diagram shows, many of these products and features have converged onto single platforms. There are now products on the market which contain all these features, plus additional capabilities. Clearly the term "Database Activity Monitoring" only covers a subset of what these tools offer. So we needed a new name to better reflect the capabilities of these technologies.

As we looked deeper we realized how unusual standalone DAM products were (and still are). It gradually became clear that we were watching the creation of a platform, rather than the development of a single-purpose product. We believe the majority of database security capabilities will be delivered either as a feature of a database management system, or in these security products. We have decided to call them *Database Security Platforms*, as that best reflects the current state of the market and how we see it evolving.

Some of these products include non-database features designed for data center security. We wouldn't be surprised to see this evolve into a more generic data center security play, but it's far too early to see that as a market of its own.

Market and Product Evolution

We already see products differentiating based on user requirements. Even when feature parity is almost complete between products, we sometimes see vendors shifting them between different market sectors. We see primary use cases, and we expect products to differentiate along these lines over time:

- **Application and Database Security**: These products focus more on integrating with Web Application Firewalls and other application security tools. They place a higher priority on vulnerability and exploit detection and blocking; and sell more directly to security, application, and database teams.
- **Data and Data Center Security**: These products take a more data-center-centric view of security. Their capabilities will expand more into data server security and integrity, unstructured file security and they will focus more on detecting and blocking security incidents. They sell to security, database, and data center teams.
- **Audit and Compliance**: Products that focus more on meeting audit requirements -- and so emphasize monitoring capabilities, user rights management, and data masking.

While there is considerable feature overlap today, we expect differentiation to increase as vendors pursue these different market segments and buying centers. Even today we see some products evolving primarily in one of these directions, which is often reflected in their sales teams and strategies.

This should give you a good idea of how we got here from the humble days of DAM, and why this is more than just a rebranding exercise. We don't know of any DAM-only tools left on the market, so that name clearly no longer fits. As a user and/or buyer we also think it's important to know which combination of features to look at, and how they can indicate the future of your product. Without revisiting the lessons learned from other security platforms, suffice it to say that you will want a sense of which paths the vendor is heading down before locking yourself into a product that might not meet your needs in 3-5 years.

Use Cases

Database Security Platforms are incredibly versatile -- offering benefits for security, compliance, and even operations. The following are some classic use cases and ways we often see them used:

Monitoring and assessment for regulatory compliance

Traditionally the biggest driver for purchasing a DAM/DSP product was to assist with compliance, with Sarbanes-Oxley (SOX) almost single-handedly driving the early market. The features were mostly used in for compliance in a few particular ways:

- To assess in-scope databases for known security issues and policy compliance. Some regulations require periodic database assessment for security issues, policy (configuration) compliance, or both.
- To assess databases for entitlement issues related to regulatory compliance. While all vulnerability tools can assess database platforms to some degree, no non-database-specific tools can perform credentialed scanning and assessment of user entitlements. This is now often required by certain regulations to ensure users cannot operate outside their designated scope, and to catch issues like users assigned multiple roles which create a conflict of interest. This can be evaluated manually, but it is far more efficient to use a tool if one is available.
- To monitor privileged users. This is often the single largest reason to use a DSP product in a compliance project. Logon/logoff reports as well as tracking all system activity, and alerting when administrators and select privileged users view sensitive information.

- For comprehensive compliance reports spanning multiple databases and applications. Policy-level reports demonstrate that controls are in place, while other reports provide the audit trail necessary to validate the control. Most tools include such reports for a variety of major regulations, with tailored formats by industry.

Web application security

SQL Injection is an attack against a database. While the attack works *through an application,* it is a database attack. And it's remained one of the three most common attack vectors for the last decade. Almost all web applications are backed by databases, but since it's the application that faces the public, databases are often an afterthought and left unprotected by application developers. Web Applications Firewalls can block some SQL injection, but a key limitation is that they don't necessarily understand the database they are protecting, and so are prone false positives and negatives.

DSPs provide a similar capability -- at least for database attacks -- but with detailed knowledge of both the database type and how the application uses it. For example, if a web application typically queries a database for credit card numbers, the DSP tool can generate an alert if the application requests more card numbers than a defined threshold (often 1).

A DSP tool with content analysis can do the same thing without the operator having to identify the fields containing credit card numbers. Instead you can set a generic "credit card" policy that alerts any time a credit card is returned in a query to the web application server, as nearly no front-end applications ask for full card numbers anymore -- they are typically left to transaction systems instead.

We have only scratched the surface of the potential security benefits for web apps. For example, query whitelisting can alert any time new queries or patterns appear. It is increasingly common for attackers to inject or alter stored procedures in order to take control of databases, and stored procedure monitoring picks up attacks that a WAF might miss.

Some tools on the market even communicate violations back to a WAF, either for alerting or to terminate suspicious sessions and even block the offending IP address.

Change management

Critical databases go down more often due to poor change management than due to attacks. Unlike application code changes, administrators commonly jump right into production databases and directly manipulate data in ways that can easily cause outages.

Adding closed-loop change management supported by DSP reduces the likelihood of a bad change, and provides much deeper accountability -- even if shared credentials are used. Every administrator action in the database can be tracked and correlated back to a specific change ticket, with monitoring showing the full log of every SQL command -- and often return values as well.

Legacy system and service account support

Many older databases have terrible logging and auditing features that can crush database performance, when they are even available. Such older databases are also likely to include poorly secured service accounts (although we concede that stored plain-text credentials for application accounts are still all too common in general). DSP can generate an audit trail where the database itself does not offer one, and DSP tools tend to support older databases -- even those no longer supported by the database vendor. Even modern databases with auditing tend to impose a greater performance impact than DSPs.

They can also audit service accounts -- generic accounts used by applications to speed up performance -- and even alert on unusual activity. This can be especially useful with even a simple rule -- such as alerting on any access attempt using service account credentials from anywhere other than the application server's IP address.

We've received several comments during the course of this research project that security is where the growth in demand is coming from, not compliance or operations. Several mentioned that, outside of large enterprise, web facing database and application protection now outweighed compliance. What this means for most buyers is DSP vendors are putting more resources behind application security features in response to market demand. Our research confirms a growth in security-first DSP deployments, especially in the area of protecting legacy code that have web interfaces bolted on. However this remains a small percentage of the overall market from our customer interviews. Most buyers use DSP to fulfill compliance obligations first, and then leverage the investment for operational efficiency and security second. It's a testament to platform maturity that these products are able to cover each of the use cases, and provide advanced features for each audience.

Data and Event Collection

Central to the evolutionary theme of activity monitoring is the migration of the event collection mechanisms from native audit, to monitoring network activity, to agent-based activity monitoring. These are all database-specific information sources. That's important, because what you can do is highly dependent on the data you can collect. For example, the *big reason* agents are the dominant collection model is that you need them to comprehensively monitor administrators -- network monitoring can't do that (and is quite difficult in distributed environments).

The development of DAM into DSP also entails examination of a broader set of application-related events. By augmenting the data collection agents we can see other applications in *addition to databases* -- even including file activity. This means that it has become possible to monitor SAP and Oracle application events -- in real time. It's possible to monitor user activity in a Microsoft SharePoint environment, regardless of how data is stored. We can even monitor file-based non-relational databases. We can perform OS, application, and database assessments through the same system.

A slight increase in the scope of data collection means much broader application-layer support. Not that you necessarily need it -- sometimes you want a narrow database focus, while other times you will need to cast a wider net. We will describe all the options to help you decide which best meets your needs.

Let's take a look at some of the core data collection methods used by customers today:

Event Sources

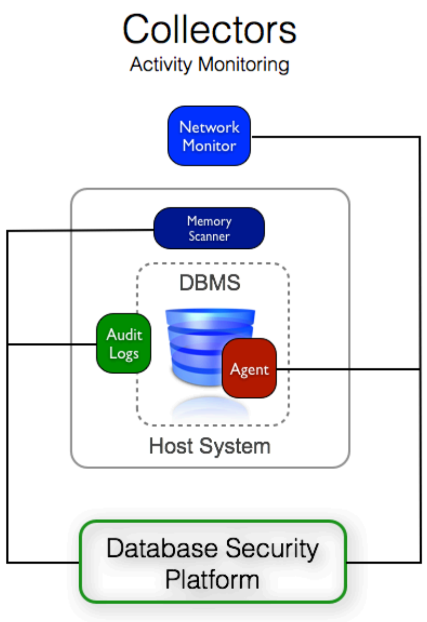
Local OS/Protocol Stack Agents

A software 'agent' is installed on the database server to capture SQL statements as they are sent to the databases. The events captured are returned to the remote Database Security Platform. Events are optionally inspected by the local agent for real-time analysis and response. The agents are either deployed into the host's network protocol stack or embedded into the operating system, to capture communications to and from the database. They see all external SQL queries sent to the database, including their parameters, as well as query results. Most critically, they should capture administrative activity from the console that does not come through normal network connections. Some agents provide an option to block malicious activity -- usually by dropping the query rather than transmitting it to the database, or by resetting the suspect user's database connection.

Agents that embed into the OS offer the advantage of full session visibility, at the cost of requiring a system reboot during installation. Non-kernel agents do not incur this one-time penalty. Early implementations struggled with reliability and platform support problems, causing system hangs, but these issues are now fortunately rare. Current implementations tend to be reliable, with low overhead and good visibility into database activity. Agents are a basic requirement for any DSP solution, as they are a relatively low-impact way of capturing all SQL statements -- including those originating from the console and arriving via encrypted network connections.

Performance impact these days is very limited, but you will still want to test before deploying into production.

Network Monitoring



An exceptionally low-impact method of monitoring SQL statements sent to the database. By monitoring the subnet (via network mirror ports or taps) statements intended for a database platform are 'sniffed' directly from the network. This method captures the original statement, the parameters, the returned status code, and any data returned as part of the query operation. All collected events are returned to a server for analysis. Network monitoring has the least impact on the database platform and remains popular for monitoring less critical databases, where capturing console activity is not required.

Lately the line between network monitoring capabilities and local agents that just collect network activity has blurred. Network monitoring is now commonly deployed via a local agent monitoring network traffic on the database server itself, thereby enabling monitoring of encrypted traffic. Some of these 'network' monitors still miss console activity -- specifically privileged user activity -- making them inappropriate for meeting most compliance mandates. Further, the introduction of an agent also incurs some performance penalty on the database server. Users must also verify that the monitor is collecting database response codes, and should determine exactly which local events are captured, during the evaluation process.

Memory Scanning Agents

Memory scanners read the active memory structures of a database engine, monitoring new queries as they are processed. Deployed as an agent on the database platform, the memory scanning agent activates at pre-determined intervals to scan for SQL statements. Most memory scanners immediately analyze queries for policy violations -- even blocking malicious queries -- before returning results to a central management server. There are numerous advantages to memory scanning, as these tools see *every* database operation, including all stored procedure execution. Additionally, they do not interfere with database operations.

You'll need to be careful when selecting a memory scanning product -- the quality of the different implementations varies. Some vendors only support memory scanning on select platforms -- and not across all databases. Some vendors don't capture query variables -- only the query structure -- limiting the usefulness of their data. And some vendors still struggle with performance, occasionally missing queries. When properly implemented, memory scanners are an excellent, enterprise-ready option for monitoring events and enforcing policy with very low performance impact.

Database Audit Logs

Database Audit Logs are still commonly used to collect database events. Most databases have native auditing features built in; they can be configured to generate an audit trail that includes system events, transactional events, user events, and other data definitions not available from any other sources. The stream of data is typically sent to one or more locations assigned by the database platform, either in a file or *within* the database itself. Logging can be implemented through an agent, or logs can be queried remotely from the DSP platform using SQL.

Audit logs are preferred by some organization because they provide a series of database events *from the perspective of the database*. The audit trail reconciles database rollbacks, errors, and uncommitted statements -- producing an accurate representation of changes made. But the downsides are equally serious. Historically, audit performance was horrible. While the database vendors have improved audit performance and capabilities, and DSP vendors provide great advice for tuning audit trails, bias against native auditing persists. And frankly, it's easy to mess up audit configurations. Additionally, the audit trail is not really intended to collect `SELECT` statements -- viewing data -- but focused on changes to data or the database system. Finally, as the audit trail is stored and managed on the database platform, it competes heavily for database resources -- much more than other data collection methods. But given the accuracy of this data, and its ability to collect internal database events not available to network and OS agent options, audit remains a viable -- if not essential -- event collection option.

There are *many* more methods of gathering data and events, but we're focusing on the most commonly used. If you are interested in a more depth on the available options, our blog post on Database Activity Monitoring & Event Collection Options provides much greater detail.

Discovery and Assessment Sources

Network Scans: Most DSP platforms offer database discovery capabilities, either through passive network monitoring for SQL queries or through active TCP scans of open database ports. Some databases reveal database type, revision and patch information at the port. As many customers now view this as potentially supplying vulnerability data to potential attackers, most database vendors limit the information provided by the port, and most customers alter the default port number to avoid detection.

System Tables: Databases store configuration, patch, user and feature information internally. Assessment solutions use database credentials to look inside the database and gather configuration data that is simply not available when examining a database externally. Most customers use remote credentialed scanning of internal database structures for *data* discovery, user entitlement reporting, and configuration assessment. Some assessment tools will examine configuration files, but this option is increasingly rare.

None of these capabilities are new, but remote scanning with read-only user credentials is the the standard data collection method for preventative security controls.

Expanded Collection Sources

A couple new features broaden the focus of DAM. Here's what's new:

File-Based Activity

One of the most intriguing recent changes in event monitoring has been the collection of file activity. Some DSP products now support file-based database systems, which are increasingly common due to the rise of NoSQL and Big Data. Other products include complete File Activity Monitoring capabilities (which we discuss later).

This evolution is important for two reasons. The first is that document and data management systems are moving away from strictly relational databases as the storage engine of choice. Microsoft SharePoint, mentioned above, is a hybrid of file management and relational data storage. File monitoring provides a means to monitor document usage and alert on policy violations. Some customers need to address compliance and security issues consistently, and don't want to differentiate based on the idiosyncrasies of underlying storage engines, so file monitoring event collection offers consistent data usage monitoring.

Another interesting aspect of file monitoring is that most of the databases used for Big Data are non-relational file-based data stores. Data elements are self-describing and self-indexing files. File monitoring is needed to support these systems.

Application Monitoring

Databases are used to store application data and persist application state. It's almost impossible to find a database *not* serving an application, and equally difficult to find an application that does not use a database. As a result monitoring the database is often considered sufficient to understand application activity. However most of you in IT know database monitoring is actually inadequate for this purpose. Applications use hundreds of database queries to support generic forms, connect to databases with generic service accounts, and/or uses native application codes to call embedded stored procedures rather than direct SQL queries. Their activity may be too generic, or inaccessible to, traditional Database Activity Monitoring solutions. We now see agents designed and deployed specifically to collect application events, rather than database events. For example SAP transaction codes can be decoded, associated with a specific application user, and then analyzed for policy violations. As with SIEM systems, much of the value comes from better linking of user identity to activities. But extending scope to embrace the application layer directly provides better visibility into application usage and enables more granular policy enforcement.

Technical Architecture and Features

One of the key strengths of DSP is its ability to scan and monitor multiple databases running on multiple database management systems (DBMSs) across multiple platforms (Windows, Unix, etc.). The DSP tool aggregates information from multiple collectors to a secure central server. In some cases the central server/management console also collects information while in other cases it serves merely as a repository for data from collectors.

Base Architecture

This creates three options for deployment, depending on organizational requirements:

- *Single Server/Appliance*: A single server, appliance, or software agent serves as both the sensor/collection point and management console. This mode is typically used for smaller deployments.
- *Two-tier Architecture*: This option consists of a central management server and remote collection points/sensors. The central server does no direct monitoring, but aggregates information from remote systems, manages policies, and generates alerts. Some perform assessment functions directly. The remote collectors can use *any* of the collection techniques.
- *Hierarchical Architecture*: Collection points/sensors/scanners aggregate to business-level or geographically distributed management servers, which in turn report to an enterprise management server. Hierarchical deployments are best suited for large enterprises, which often have different business, compliance or geographic needs. They can also be configured to only pass certain kinds of data between the tiers to manage large volumes of information or maintain unit/geographic privacy, and to satisfy policy requirements.

This can be confusing because each server or appliance can manage multiple assessment scanners, network collectors, or agent-based collectors can perform some monitoring directly. But a typical deployment includes a central management server (or cluster) handling all the management functions, with collectors spread out to handle activity monitoring on the databases.

Blocking Options

There are two different ways to block queries, depending on your deployment architecture and choice of collection agents.

- *Agent-based Blocking*: The software agent is able to directly block queries -- the actual technique varies with the vendor's agent implementation. Agents can block inbound queries, returned results, or both.
- *In-line bridging*: A bridge is a simple, non-IP addressable service that moves packets from network adapter to another, but allows for the inspection of packets as they pass through the bridge. Advantages include real-time analysis and blocking, the option to “fail open” in the event the filter fails, and minimal changes to data center infrastructure.

- *Proxy-based Blocking*: Instead of connecting directly to the database, all connections are to a local or network-based proxy (which can be a separate server/appliance or local software). The proxy analyzes queries before passing them to the database, and can block by policy.

We will go into more detail on blocking later in this series, but the important point is that if you want to block, you need to either deploy some sort of software agent or proxy the database connection. Further, it's critical to structure policies in such a way to reduce false positives. Next we will recap the core features of DAM and show the subtle additions to DSP.

Core Features

We have covered the basics of how a Database Security Platform is architected and deployed, now it's time to go into greater detail as to what they *do*. The following short list are the most important features common to all Database Security Platforms, and the primary reasons to buy these systems.

Activity Monitoring

The single defining feature of Database Security Platforms is their ability to collect and monitor *all* database activity. This includes all administrator and system activity that touches data (short of things like indexing and other autonomous internal functions).

We have already covered the various event sources and collection techniques used to power this monitoring, but let's briefly review what kinds of activity these products can monitor:

A key way to determine if a product fully monitors a database is to ask if it tracks **SELECT** queries. The only way to do this is to see the actual SQL.

- **All SQL** -- DML, DDL, DCL, and TCL: Activity monitoring needs to include all interactions with the data in the database, which for most databases (even non-relational) involves some form of SQL (Structured Query Language). SQL breaks down into the Data Manipulation Language (DML, for select/update queries), the Data Definition Language (DDL, for creating and changing table structure), the Data Control Language (DCL, for managing permissions and such) and the Transaction Control Language (TCL, for things like rollbacks and commits). As you likely garnered from our discussion of event sources, depending on a product's collection techniques, it may or may not cover all this activity.
- **SELECT` queries**: Although a `SELECT` query is merely one of the DML activities, due to the potential for data leakage, `SELECT` statements are monitored particularly closely for misuse. Common controls examine the type of data being requested and the size of the result set, and check for SQL injection.
- **'Sensitive Data' access**: Technically a subset of the 'SELECT' queries listed above, policies are specifically focused on the access of tables or columns with sensitive data such as credit cards or social security numbers.
- **Administrator activity**: Most administrator activity is handled via queries, but administrators have a wider range of ways they can connect to database than regular users, and more ability to hide or erase traces of their activity. This is one of the biggest reasons to consider a DSP tool, rather than relying on native auditing.
- **Stored procedures, scripts, and code**: Stored procedures and other forms of database scripting are used in attacks to circumvent user-based monitoring controls. DSP tools should also track internal activity (if necessary).
- **File activity**, if necessary: While a traditional relational database relies on query activity to view and modify data, many newer systems (and a few old ones) work by manipulating files directly. If you can modify the data by skipping the Database Management System and editing files directly on disk (without breaking everything, which happens with most relational systems), some level of monitoring is called for.

Even with a DSP tool, it isn't always viable to collect *everything*, so the product should support custom monitoring policies to select what types of activities and/or user accounts to monitor. For example, many customers deploy a tool only to monitor administrator activity, or to monitor all administrators' `SELECT` queries and all updates by everyone.

Policy Enforcement
One of the distinguishing characteristics of DSP tools is that they don't just collect and log activity -- they analyze it in real or near-real time for policy violations. While still technically a detective control (we will discuss preventative deployments later), the ability to alert and respond in or close to real time offers security capabilities far beyond simple log analysis. Successful database attacks are rarely the result of a single malicious query -- they involve a sequence of events (such as exploits, alterations, and probing) leading to eventual damage. Ideally, policies are established to detect such activity early enough to prevent the final loss-bearing act. Even when an alert is triggered after the fact, it facilitates immediate incident response, and investigation can begin immediately rather than after days or weeks of analysis.

Monitoring policies fall into three basic categories:

- **Attribute-based**: Rules are established based upon common use cases and monitored for violation. The rules are based upon query attributes like time of day, user, application, location, etc. They can include specific table references, result counts, administrative functions (such as new user creation and rights changes), signature-based SQL injection detection, `UPDATE` or other transactions by users of a certain level on certain tables/fields, or any other activity that can be specifically described. Advanced rules can correlate across different parts of a database or even different databases, accounting for data sensitivity based on DBMS labels or during policy creation. While attribute-based detection remains a 'bread-and-butter' approach for general polices, deficiencies in behavioral and attack detection resulted in the need for alternative analysis methods.
- **Heuristic**: Monitoring database activity builds a profile of 'normal' activity (we also call this "behavioral profiling"). Deviations then generate policy alerts. Heuristics are complicated and require tuning to work effectively. They are a good way to build a base policy set, especially with complex systems where manually creating deterministic rules by hand isn't realistic. Policies are -- in most cases automatically -- tuned over time to reduce false positives. For well-defined systems where activity is consistent, such as an application talking to a database using a limited set of queries, they are very useful. Of course heuristics fail when malicious activity is mis-profiled as good activity, and do a poor job with database attacks like SQL Injection.
- **Lexical**: Mostly used for blocking, lexical analysis of the query *structure* enables easier detection of SQL Injection and similar attacks. By examining the elements in the FROM and WHERE clauses of the query, or whitelisting a subset of the entire SQL grammar (e.g. a list of all *possible* queries), any request that does not match an known query 'silhouette' is blocked. Lexical analysis is an excellent way to keep false positives low while greatly improving attack detection, however they require updates each time the calling application is modified with new queries. Further, lexical analysis is not ideal for 'fuzzier' policies, such as detection of odd user behavior.

The market started with attribute based detection, but these policies lacked the capability to adjust to the environment. They were unable to determine of any given request was asking for "too much data", or if a user started issuing unusual -- albeit legal -- queries. The result was heuristics based analysis to provide behavioral identification techniques to policy enforcement. But this still left a considerable gap in attack detection, and customers quickly found 'false positive' rates were far too high, which is of critical importance if your goal is to block malicious queries. It's one thing to receive bogus alerts you need to comb through, it's quite another to halt legitimate queries from an application, which result in severe side-effects or application failures. The result was lexical analysis capabilities, which provides the capability to categorize queries based upon their structure, and accurately identify approved queries. The result was 'query white listing' known sets of valid queries, and 'virtual patching' (i.e. black-listing) option to block exploits in the event patches cannot be deployed in a timely fashion

Aggregation and Correlation

One characteristic which Database Security Platforms share with System Information and Event Management (SIEM) tools is their ability to collect disparate activity logs from a variety of database management systems -- and then to aggregate, correlate, and enrich event data. The combination of multiple data sources across heterogenous database types enables more complete analysis of activity rather than working only on one isolated query at a time. And by understanding the Structured Query Language (SQL) syntax of each database platform, DSP can interpret queries and parse their meaning. While a simple `SELECT` statement might mean the same thing across different database platforms, each database management system (DBMS) is chock full of its own particular syntax. A DSP solution should understand the SQL for each covered platform and be able to normalize events so the analyst doesn't need to know the ins and outs of each DBMS. For example, if you want to review all privilege escalations on all covered systems, a DSP tool will recognize those various events across platforms and present you with a complete report without you having to understand the SQL particulars of each one.

Assessment

We typically see three types of assessment included in DSP products:

- Vulnerability assessments: Credentialed and non-credentialed scans of databases to identify known vulnerabilities, missing security patches, default accounts, suspect or unused database features, default/weak passwords, and many more security and compliance related checks.
- Configuration assessments: Credentialed and non-credentialed analysis of database internal and external configurations (i.e. settings at file system layer), for both security-related issues and configuration policy compliance.
- Entitlement analysis: Credentialed scan of user accounts within a database, primarily to identify security and compliance issues, but also often used for general account cleanup. Use of default accounts, roles for column and schema access, or users belonging to admin and user roles just to name a few. Database user accounts are often more complex than those of other systems due to the various ways user accounts are defined -- for example, it's common to have one human with a network user, database user, and system user accounts. Permissions are further complicated by group and role based entitlements the database provides.

These tools are typically bundled with **database discovery** capabilities to find databases on the network to assess, which we cover later in this section.

Extended Features

Extended features covers extensions of the core features, focusing on new methods of data analysis and protection and with several operational capabilities needed for enterprise deployments. A key area where DSP extends DAM is in novel security features to protect databases and extend protection across other applications and data storage repositories.

In other words, these are some of the big differentiating features that affect which products you look at if you want anything beyond the basics, but they aren't all in wide use.

Analysis and Protection

Blocking/Virtual Patching

Most DSP platforms have the ability to block inbound queries by acting as a 'Database Firewall', halting queries before they execute. There are several methods available to customers, including in-line appliances, network connection resets and memory scanning agents. A specific type of blocking called 'Virtual Patching', where queries that exercise known exploits are blocked before patches are deployed is common amongst DSP providers. We did not include this as a core DSP feature for the reasons that adoption remains restricted to a small number of the total databases being monitored, and the effectiveness of the blocking is highly dependent upon additional capabilities the vendor provides. For example, for 'Virtual Patching' to be successful, the vendor needs to provide timely and effective attack signatures. Blocking in general is more effective when coupled with lexical analysis as it improves detection rates while keeping false-positives low. Database Firewalls should not be confused with Web Application Firewalls which we will discuss later.

Query Whitelisting

Query 'whitelisting' is where the DSP platform only permits known SQL queries to run on the database. This is a form of blocking, as we discussed in the base architecture section. But traditional blocking techniques rely on query parameter and attribute analysis. This technique is based upon lexical analysis, and has two significant advantages. First is that detection is based on the *structure* of the query, matching the format of the `FROM` and `WHERE` clauses, to determine if the query matches the approved list. Second is how the list of approved queries is generated. In most cases the DSP maps out the entire SQL grammar -- in essence a list of every possible supported query -- into binary search tree for super fast comparison. Alternatively, by monitoring application activity, the DSP platform can automatically mark which queries are *permitted* in baselining mode -- of course the user can edit this list as needed. Any query *not* on the white list is logged and discarded -- and never reaches the database. With this method of blocking false positives are very low and the majority of SQL injection attacks are automatically blocked. The downside is that the list of acceptable queries must be updated with each application change -- otherwise legitimate requests are blocked.

Dynamic Data Masking

Masking is a method of altering data so that the original data is obfuscated but the aggregate *value* is maintained. Essentially we substitute out individual bits of sensitive data and replace them with random values that look like the originals. For example we can substitute a list of customer names in a database with a random selection of names from a phone book. Several DSP platforms provide on-the-fly masking for sensitive data. Others detect and substitute sensitive information prior to insertion. There are several variations, each offering different security and performance benefits. This is different from the dedicated static data masking tools used to develop test and development databases from production systems.

Application Activity Monitoring

Databases rarely exist in isolation -- more often they are extensions of applications, but we tend to look at them as isolated components. Application Activity Monitoring adds the ability to watch application activity -- not only the database queries that result from it. This information can be correlated between the application and the database to gain a clear picture of just how data is used at both levels, and to identify anomalies which indicate a security or compliance failure. There are two variations currently available on the market. The first is Web Application Firewalls, which protect applications from SQL injection, scripting, and other attacks on the application and/or database. WAFs are commonly used to monitor application traffic, but can be deployed in-line or out-of-band to block or reset connections, respectively. Some WAFs can integrate with DSPs to correlate activity between the two. The other form is monitoring of application specific events, such as SAP transaction codes, as the application communicates with the database. Some of these commands are evaluated by the application, using application logic in the database. In either case inspection of these events is performed in a single location, with alerts on odd behavior.

File Activity Monitoring

Like DAM, FAM monitors and records all activity within designated file repositories at the user level and alerts on policy violations. Rather than `SELECT`, `INSERT`, `UPDATE`, and `DELETE` queries, FAM records file opens, saves, deletions, and copies. For both security and compliance, this means you no longer care if data is structured or unstructured -- you can define a consistent set of policies around *data*, not just *database*, usage. You can read more about FAM in Understanding and Selecting a File Activity Monitoring Solution.

Query Rewrites
Another useful technique for protecting data and databases from malicious queries is query rewriting. Deployed through a reverse database proxy, incoming queries are evaluated for common attributes and query structure. If a query looks suspicious, or violates security policy, it is substituted with a similar *authorized* query. For example, a query that includes a column of Social Security numbers can be omitted from the results by removing that portion of the `FROM` clause. Queries that include the highly suspect `"1=1"` in the `WHERE` clause would then simply return the value `1`. Rewriting queries protects application continuity, as the queries are not simply discarded -- they return a subset of the requested data, so false positives don't cause the application to hang or crash.

Connection-Pooled User Identification
One of the problems with connection pooling, whereby an application using a single shared database connection for all users, is loss of the ability to track which actions are taken by which users at the database level. Connection pooling is common and essential for application performance, but if all queries originate from the same account that makes granular security monitoring difficult. This feature uses a variety of techniques to match each query with an application user, both enriching audit data and providing a means to implement user specific monitoring policies.

Discovery and Content Analysis

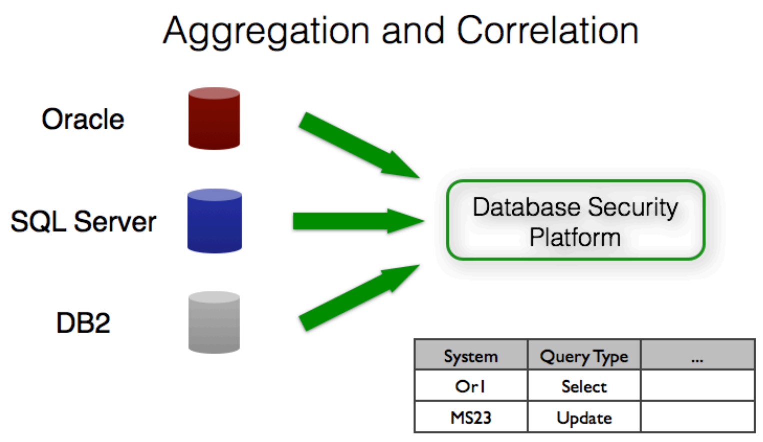
Database Discovery
Databases have a habit of popping up all over the place without administrators being aware. Everything from virtual copies of production databases showing up in test environments, to Microsoft Access databases embedded in applications. These databases are commonly not secured to any standard, often have default configurations, and provide targets of opportunity for attackers. Database discovery works in one of two ways: active scanning and passive detection. Most tools perform an active scan of networks, trying to identify databases by interrogating a database port. Commonly there is an option for quick scans -- just looking for databases on standard port numbers -- or complete scans that iterate through the entire port address range. The later is time consuming, but as it's a recommended best security practice to run databases on non-standard ports, this option is critical to success. The second option is to listen on the network, in promiscuous mode, looking for anyone communicating with a database. This passive discovery mode takes more time to discover databases and only works on the local network 'sub-net'. It does however offer two significant advantages that it results in *far* less network overhead, and it does not cause other security tools (e.g. IDS, NAC, SIEM) to alert/halt suspect network activity. Discovery tools can snapshot all current databases and alert admins when new undocumented databases appear. In some cases they can automatically initiate a vulnerability scan.

Content Discovery
As much as we like to think we know our databases, we don't always know what's inside them. DSP solutions offer content discovery features to identify the use of things like Social Security numbers, even if they aren't located where you expect. Discovery tools crawl through registered databases, looking for content and metadata that match policies, and generate alerts for sensitive content in unapproved locations. For example, creating a policy to identify credit card numbers in any database and generate a report for PCI compliance. The tools can run on a scheduled basis so you can perform ongoing assessments, rather than combing through everything by hand every time an auditor comes knocking. Most start with a scan of column and table metadata, then follow with an analysis of the first ***n*** rows of each table, rather than trying to scan *everything*.

Dynamic Content Analysis

Some tools allow you to act on the discovery results. Instead of manually identifying every field with Social Security numbers and building a different protection policy for each location, you create a single policy that alerts every time an administrator runs a `SELECT` query on *any* field discovered to contain one or more SSNs. As systems grow and change over time, the discovery continually identifies fields containing protected content and automatically applies the policy. We are also seeing DSP tools that monitor the results of live queries for sensitive data. Policies are then freed from being tied to specific fields, and can generate alerts or perform enforcement actions based on the result set. For example, a policy that generates an alert any time a query result contains a credit card number, no matter what columns were referenced in the query.

Administration



System	Query Type	...
Orl	Select	
MS23	Update	

Conceptually DSP is pretty simple: collect data from databases, analyze it according to established rules, and react when a rule has been violated. The administrative component of every DSP platform is designed to follow these three basic tasks: data management, policy management, and workflow management. In addition to these three basic functions, we also need to administer the platform itself, as we do with any other application platform.

As we described in the earlier section on DSP technical architecture, all events are fed to a central server. The DAM precursors evolved from single servers, to two-tiered architectures, and finally into a hierarchal services in order to scale up to enterprise environments. The good news is that administration is inherently centralized, with system maintenance, data storage, and policy management all available from a single console. While administration is now usually through a browser, the web application server that handles management sessions is built into the central management server. Unlike some other security products, not much glue code or browser tricks is required to stitch things together.

System Management

- *User Management*: With access to many different databases, most filtering and reporting on sensitive data, user management is critical for security. Establishing who can make changes to policies, read collected data, or administer the platform are all specialized tasks, and these groups of users are typically kept separate. All DSP solutions offer different methods for segregating users into different groups, each with differing granularity. Most of the platforms offer integration with directory services to aid in user provisioning and assignment of roles.
- *Collector/Sensor/Target Database Management*: Agents and data collectors are managed from the central server. While data and policies are stored centrally, the collectors -- which often enforce policy on the remote database -- must periodically synch with the central server to update rules and settings. Some systems require the administrator to 'push' rules out to agents or remote servers, while others synch automatically.
- DSP System Management: DSP is, in and of itself, and application platform. It has web interfaces, automated services, and databases like most enterprise applications. As such it requires some tweaking, patching, and configuration to perform its best. For example, the supporting database often needs pruning to clear out older data, vendor supplied vulnerability assessment rules require updates, and the system may need additional resources for data storage and reports. The system management interface is provided via a web browser, but only available to authorized administrators.

Data Aggregation and Correlation

The one characteristic Database Activity Monitoring solutions share with log management, and even Security Information and Event Management, tools is their ability to collect disparate activity logs from a variety of database management systems. They tend to exceed the capabilities of related technologies in their ability to go "up the stack" in order to gather deeper database activity application layer data, and in their ability to correlate information. Like SIEM, DSP aggregates, normalizes, and correlates events across many heterogenous sources. Some platforms even provide an optional 'enrichment' capability by linking audit, identity and assessment data to event records. For example, providing both 'before' and 'after' data values for a suspect query.

Despite central management and correlation features, the similarities with SIEM end there. By understanding the Structured Query Language (SQL) of each database platform, these platforms can interpret queries and understand their meaning. While a simple SELECT statement might mean the same thing across different database platforms, each database management system (DBMS) is full of its own particular syntax. DSP understands the SQL for each platform is able to normalize events so the user doesn't need to know the ins and outs of each DBMS. For example, if you want to review all privilege escalations on all covered systems, a DAM solution will recognize those events, regardless of platform and present a complete report, without you having to understand the SQL.

A more advanced feature is to then correlate activity across different transactions and platforms, rather than looking only at single events. For example, some platforms recognize a higher than normal transaction volume by a particular user, or (as we'll consider in policies) can link a privilege escalation event with a large `SELECT` query on sensitive data, which indicates an attack. All activity is also centrally collected in a secure repository to prevent tampering or a breach of the repository itself. Since they collect massive amounts of data, DSPs must support automatic archiving. Archiving should support separate backups of system activity, configuration, policies, alerts, and case management; and encrypt under separate keys to support separation of duties.

Policy Management

All platforms come with sets of pre-packaged policies for security and compliance. For example, every product contains hundreds, if not thousands, of assessment policies that identify vulnerabilities. Most platforms come with pre-defined policies for monitoring standard deployments of databases behind major applications such as Oracle Financials and SAP. Built-in policies for PCI, SOX, and other generic compliance requirements are also available to help you jump-start the process and save many hours of policy building. Every single policy has the built-in capability of generating an alert if the rule is violated. Note that every user needs to tune or customize a subset of pre-existing policies to match their environment, and create others to address specific risks to their data. They are still far better than starting from scratch.

Policies for activity monitoring include user/group, time of day, source/destination, and other important contextual options. The vendor should offer different analysis techniques based on attributes, heuristics, context, and content analysis. Support for advanced definitions, such as complex multi-level nesting and combinations of statements is something you should look for. If a policy violation occurs you can specify any number of alerting, event handling and reactive actions. Ideally, the platform will include policy creation tools that limit the need to write everything out in SQL or some other definition language; it's much better if your compliance team does not need to learn SQL programming to create policies. You can't avoid having to do some things by hand, but policy management should be as point-and-click easy as possible.

For common types of policies, including detecting privileged user activity and quantity thresholds on sensitive data, policy wizards are extremely useful. One of the distinguishing characteristics of activity monitoring tools is that they don't just collect and log activity -- they analyze it in real or near-real time for policy violations. While still technically a detective control (we will talk about preventative policies later), the ability to alert and respond in practically real time offers security capabilities far beyond simple log analysis. Successful loss-bearing database attacks are rarely the result of a single malicious query -- they involve a sequence of events leading to the eventual damage. Ideally, policies will be established to detect the activity early enough to prevent the final loss-bearing act. Even alerts triggered after the fact facilitate immediate incident response, and investigation can begin immediately, rather than after days or weeks of analysis. Policy wizards are extremely useful -- especially for common policy types, such as detecting privileged user activity and count thresholds on sensitive data.

Policies for assessments are just as critical as monitoring policies, but they are handled differently. Unlike monitoring policies, most assessment policies will be provided by the vendor. New database vulnerabilities are discovered all the time, and to keep pace DSP vendors need to develop and distribute new policies to address new threats. Most vendors provide new policies on a weekly or monthly basis. Unlike monitoring policies, vendors typically do not share the underlying SQL with customers. The code to detect vulnerabilities, as well as the vulnerability research data, is considered "secret sauce". If you want to customize vulnerability scans you will either need to develop your own or work with your vendor. Policy management interfaces for assessment and monitoring are both available through web interfaces, but not typically integrated -- both because the underlying code that supports the policies is entirely different, and to provide separation of duties between monitoring and assessment audiences.

With policy management we are seeing a trend with DSP where the platform provides an additional policy management UI targeted specifically at non-database or non-technical stakeholders. The goal is to offer a gateway for people who don't necessarily understand SQL to administer policies without forcing them to understand technical intricacies of the database system. These new policy management interfaces are designed for compliance, operations and security groups to implement controls, offering an abstract view of database security. The DSP platform automatically maps the policy to the specific database rule(s) that implement the control. These capabilities are fairly new, which shows in the quality and flexibility of these interfaces, but we expect continued product maturity in this area.

Reports and Workflow

Reports

As with nearly any security tool, you'll want flexible reporting options, but pay particular attention to compliance and auditing reports to support compliance needs. Aside from all the security advantages we've been talking about, many organizations initially deploy monitoring to meet database audit and compliance requirements. Built-in report templates are universally available, and save valuable time in getting your product deployed. Some vendors work directly with auditors from the major firms to help design reports for specific regulations like SOX.

Reports fall into at least three broad categories: compliance and non-technical reports, security reports (incidents), and general technical reports.

Change management

Although many organizations have rigorous change management policies for the database platform and underlying system, far fewer enforce change management at the query level (which is invisible to traditional change management tools). Some DSP solutions offer integration with external change management and workflow tools for closed-looped tracking of query-level changes. The requested change is approved in the change management system and a ticket number issued. The DBA enters that ticket number as part of their session, and all database changes (even to individual field updates) are recorded and correlated back to the original change ticket. Changes without associated tickets are logged in compliance reports and may trigger a security alert, depending on supporting policies.

Integration

DSPs collect a wealth of information from databases that, due in part to the way data is collected, is not available to other systems. Enterprise DSP deployments often 'react' to events by streaming data to other security services. Vulnerability reports and alerts on suspicious activity are often formatted and sent to SIEM and Log Management platforms. In many cases customers have databases disconnect active sessions and lock user accounts based on perceived threats. DSP handles real time alerts and maintains its own event repository, but works in tandem with these other systems to help both near-real-time reaction and forensic auditing.

DSP Selection Process

The goal of this paper is to help consumers understand the capabilities of Database Security Platforms and how they help solve security and compliance requirements. But we're aware that many of the questions we receive come from people who need help in making a purchasing decision. They're trying to figure out what product is right for them. There are a lot of facets to DSP, so what is important to verify, what questions should be asked, and what are the potential pitfalls? And every potential customer brings a list of requirements to meet, a preference for the way they want to operate, and specific systems they need to integrate with. The following selection process guide will help you uncover the

Define Needs

Before you start looking at any tools, you need to understand why you need DSP and how you plan on using it; this includes the business processes around management, policy creation, and incident handling.

1. Create a selection committee: Database security initiatives tend to involve four major technical stakeholders, and one or two non-technical business units. On the technical side it's important to engage the database and application administrators of systems that may be within the scope of the project over time, not just those responsible for the single database and/or application you plan on starting with. Although most projects start with a limited scope -- a practice we highly recommend -- they can quickly grow into enterprise-wide programs. Security and database teams typically drive the project, and the office of the CIO is often involved due to compliance needs or to mediate cross-team issues. The technical staff will also map requirements to the technical solutions. On the non-technical side, you should have representatives from audit, business lines as well as compliance and risk (if they exist in your organization) as these teams are typically driving the requirements and supplying the budget. Once you identify the major

stakeholders, you'll want to bring representatives together into a selection committee.

2. Define the systems and platforms to protect: DSP projects are typically have multiple goals, but are driven by a clear compliance or security requirement tied to specific systems, applications, or databases. In this stage, detail the scope of what will be protected and the technical details of the platforms involved. You'll use this list to determine technical requirements and prioritize features and platform support later in the selection process. Remember that needs grow over time, so break the list into a group of high priority systems with immediate needs, and a second group summarizing platforms you may need to protect later.
3. Determine protection and compliance requirements: For some systems you might want strict preventative security controls, while for others you may just need comprehensive activity monitoring for a compliance requirement. In this step, map your protection and compliance needs to the platforms and systems from the previous step. This will help you determine everything from technical requirements to process workflow.
4. Outline process workflow and reporting requirements: Activity monitoring workflow tends to vary based on existing IT operations process, while assessment and audit have common workflows for most companies. When used as an internal control for separation of duties, security will monitor and manage events and have an escalation process should database administrators violate policy. When used as an active security control, the workflow may more actively engage security and database administration as partners in managing incidents. In most cases, audit, legal, or compliance will have at least some sort of reporting role. Policy Management, Operational Management, Reports, and Remediation are all distinct roles for most public corporations, and be reflected in your system processes and roles. Since different Database Security Platforms have different strengths and weaknesses in terms of management interfaces, reporting, and internal workflow, thinking through the process before defining technical requirements can prevent headaches down the road.

Who's in charge? One of the obstacles for successful database security is the split between database, compliance, operations and security teams. DBAs have long managed security issues for their systems, but new compliance and security demands are pulling in security professionals who have limited database skills. The key to a successful program is to get these teams working together, along with application administrators and divide up duties based on expertise and security/compliance requirements. Each needs to then share expertise for DSP projects to succeed.

By the completion of this phase you should have defined key stakeholders, convened a selection team, prioritized the systems you want to protect, determined protection requirements, and roughed out process workflow.

Formalize Requirements

This phase can be performed by a smaller team working under the mandate of the selection committee. Here, the generic needs determined in phase 1 are translated into specific technical features, while any additional requirements are considered. This is the time to come up with any criteria for directory integration, additional infrastructure integration, data storage, hierarchical deployments, change management integration, and so on. Advanced protection mechanisms can produce side effects that alter application and business logic, so it's important to document expectations in advance. You can always refine these requirements after you proceed to the selection process and get a better feel for how the products work.

At the conclusion of this stage you will have a formal RFI (Request For Information) for vendors, and a rough RFP (Request For Proposals) to clean up and formally issue in the evaluation phase.

Evaluate Products

As with any products, it's sometimes difficult to cut through the marketing materials and figure out if a product really meets your needs. The following steps should minimize your risk and help you feel confident in your final decision:

1. *Issue the RFI:* Larger organizations should issue an RFI though established channels and contact a few leading DSP vendors directly. If you're a smaller organization, start by sending your RFI to a trusted VAR and email a few of the vendors which seem appropriate for your organization. Draft the RFI based upon your requirements, avoiding generic off-the-shelf questions; you'll get better responses from the vendors.
2. *Perform a paper evaluation:* Before bringing anyone in, match any materials from the vendor or other sources to your RFI and draft RFP. Your goal is to build a short list of 3 products which match your needs. You can also use outside research sources and product comparisons.
3. *Talk to your peers:* Check with your contacts at organizations similar to yours to see if they are using a product, which ones they looked at, and why they selected the one they did.
4. *Bring in 3 vendors for on-site presentations and demonstrations:* Instead of a generic demonstration, ask each vendor to walk through your specific use cases. Don't expect a full response to your draft RFP; these meetings are to help you better understand the different options and eventually finalize your requirements.
5. *Finalize your RFP and issue it to your short list of vendors:* At this point you should completely understand your specific requirements and issue a formal, final RFP.
6. *Assess RFP responses and begin product testing:* Review the RFP results and drop anyone who doesn't meet any of your minimal requirements (such as platform support), as opposed to "nice to have" features. Then bring in any remaining products for in-house testing. We recommend that you set up a close to production test area to run scans and monitor across several databases. We encourage the use of activity playback with attacks to see if the vendor can really detect threats. You'll want to replicate your highest volume system and the corresponding traffic, if at all possible. Build a few basic policies that match your use cases, then violate them, so you can get a feel for policy creation and workflow. See the 'Internal testing' checklist later in this section for more details.

Internal Testing

In-house testing is the last chance to find problems in your selection process. Make sure you test the products as thoroughly as possible. A few key aspects to test, if you can, are:

- Platform support and installation to determine compatibility with your database/application environment. This is the single most important factor to test, including monitoring coverage for the connection methods used in your organization, since different database platforms support a variety of connection types.
- Performance. Is network or agent performance acceptable for you environment? Are there other operational considerations driving you toward one model or the other? Don't set arbitrary standards; monitor performance on your production systems to ensure your tests represent operational requirements.
- carefully consider how the software/appliance/virtual appliance with deploy your organization, including how products will be installed, configured and how they will communicate. The form factor directly impacts costs, flexibility, ease of management and scalability.
- Infrastructure requirements, meaning the what hardware systems, OS's, appliances, supporting databases or other software licensure is required? Large enterprises with existing site licenses may overlook this issue, but it becomes a serious impediment in the SMB.

- Policy creation and management. Create policies to understand the process and its complexity. Do you need to write everything as SQL? Will built-in policies meet your needs? Are there wizards and less-technical options for non-database experts to create policies? Then violate policies and try to evade or overwhelm the tool to learn its limits.
- Credential Management. How does the platform distinguish between the various user roles and does it maintain separation of duties? Does it integrate with Directory Access services?
- Incident workflow. Review the working interface with those employees who will be responsible for enforcement.
- Behavioral profiling, if the product supports it for developing policies.
- Policy updates. Can the vendor demonstrate consistent and reliable delivery of vulnerability detection policies?
- Compliance Support. Every vendor claims they have it but the quality of the support varies. Examine the assessment and monitoring policies, as well as the reports they generate. Evaluate policy management interface for customization of existing policies and the creation of new policies.
- Change management integration.
- Enforcement, blocking, rollback, and other advanced features.

Final Analysis

It's time to gather all of the data from your PoC and see how the results of your internal testing ultimately match your requirements. But it's also critical to understand the long term livability of the product. Sometimes the product easiest to set-up is not the easiest to own and operate long term. In some cases management seems trivial, but the extensive vendor support during the PoC simply created an illusion. And there is a giant gap between 'potential' and 'reality', where some systems 'theoretically' monitor the entire universe, but in practice a single DSP server will only support one large database.

- Were you able to deploy the product? If so, did you do this un-aided, or did it require assistance from the vendor?
- Did the product work as advertised? Which goals were met and which were not?
- How long did the deployment take?
- Did the platform truly support your databases?
- How hard was it to use? Did the product require up-front training or was the product intuitive enough to use without training?
- Can you maintain the platform in the long run? Is your preferred solution going to be managed in house, or is it better to have 3rd party services? How much does ongoing system maintenance weight into your decision?
- How well did the out-of-the-box policies address your needs? Were you able to create custom policies that address your organizational requirements?
- Does the product scale as advertised?
- Were false-positive and false-negative rates acceptable?
- In light of your experiences with the PoC, have you re-evaluated the total cost of deployment? The time it will take to deploy across your business? Estimates pre-PoC are often way off the mark.

Conclusion

Much in the same way Log Management tools evolved from simple event storage devices into the sophisticated SIEM platforms we see today, consuming dozens of new event types and offering many more capabilities, DAM has evolved into DSP. While DAM remains at the core of every DSP platform, today's platforms bear little resemblance to DAM of just a few years ago. Every facet -- from event collection and analysis, to policy management and enforcement -- has evolved. Audit, discovery and assessment are inherent to every platform. DSP platforms cover more types of relational databases, any may include support for application and file monitoring. Advanced analysis techniques and deployment options make blocking and policy enforcement a viable option -- as opposed to a 'marketing check-box' feature.

DSP is an extremely valuable tool for compliance and security; it is critical to the emerging practice of information-centric security. This advancement of capabilities, and the novel ways these new features are being bundled to solve security and compliance problems, warrants a fresh look at these platforms. DSP provides insight into our most sensitive systems in a non-intrusive way, and has evolved into proactive database security defense. It's one of the few tools that can immediately improve security and reduce compliance overhead without interfering with business processes.

Although deploying a tool that crosses the organizational and technical expertise boundaries between database, security, and application management can be daunting, by understanding your needs and following a structured selection process you can help ensure a successful deployment. Many clients start with a narrowly scoped project which quickly expands throughout the enterprise.

From a business standpoint, the most important factor in a successful deployment is pulling together the key stakeholders across database, security, and application administration to determine initial requirements from a technical and process standpoint. In working with hundreds of clients implementing database security, the most common obstacle I've seen is a failure to manage expectations and responsibilities across these different groups.

After level setting and determining project goals, the most important technical necessity is to understand your platform support and performance requirements. DSP offers so many capabilities that deployments invariably grow in scope and requirements. Aside from growing in terms of systems covered, programs also expand into new use cases as administrators become more comfortable with the tools and their capabilities. You may start with auditing only, then add alerting for a few basic security policies, and end with full application integration and security blocking.

Database Security Platforms are a powerful platform for compliance and information-centric security. It is relatively easy to deploy, non-intrusive, scalable, and flexible.

Who We Are

About the Authors

Adrian Lane, Analyst and CTO
Adrian Lane is a Senior Security Strategist with 25 years of industry experience, bringing over a decade of C-level executive expertise to the Securosis team. Mr. Lane specializes in database architecture and data security. With extensive experience as a member of the vendor community (including positions at Ingres and Oracle), in addition to time as an IT customer in the CIO role, Adrian brings a business-oriented perspective to security implementations. Prior to joining Securosis, Adrian was CTO at database security firm IPLocks, Vice President of Engineering at Touchpoint, and CTO of the secure payment and digital rights management firm Transactor/Brodia. Adrian also blogs for Dark Reading and is a regular contributor to Information Security Magazine. Mr. Lane is a Computer Science graduate of the University of California at Berkeley with post-graduate work in operating systems at Stanford University.

Rich Mogull, Analyst and CEO

Rich has twenty years of experience in information security, physical security, and risk management. He specializes in data security, application security, emerging security technologies, and security management. Prior to founding Securosis, Rich was a Research Vice President at Gartner on the security team where he also served as research co-chair for the Gartner Security Summit. Prior to his seven years at Gartner, Rich worked as an independent consultant, web application developer, software development manager at the University of Colorado, and systems and network administrator. Rich is the Security Editor of TidBITS, a monthly columnist for Dark Reading, and a frequent contributor to publications ranging from Information Security Magazine to Macworld. He is a frequent industry speaker at events including the RSA Security Conference and DefCon, and has spoken on every continent except Antarctica (where he's happy to speak for free — assuming travel is covered).

About Securosis

Securosis, L.L.C. is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

We provide services in four main areas:

- Publishing and speaking: Including independent objective white papers, webcasts, and in-person presentations.
- Strategic consulting for end users: Including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessments.
- Strategic consulting for vendors: Including market and product analysis and strategy, technology guidance, product evaluations, and merger and acquisition assessments.
- Investor consulting: Technical due diligence including product and market evaluations, available in conjunction with deep product assessments with our research partners.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Securosis has partnered with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis.