Understanding Comprehensive Database Security
Table of Contents

Executive Summary 4
Part 1: Database Security 6
Complexity Weakens Security 6
Time Weakens Security 7
Understanding the Need for a Formal Database Security Program 9
Three Strategic Objectives of a Formal Database Security Program 9
Seven Components of Defense in Depth 10
Part 2: People and Physical Security 30
People Security Perimeter 30
Physical and Infrastructure Security Perimeters 31
Conclusion 33
Appendix A: Tools Cited 34
Executive Summary

You Own Your Data: Protect It with a Formal Database Security Program

Organizations own and are responsible for their data, and to secure that data in a database, especially large databases supporting ERP platforms such as SAP, PeopleSoft, and the Oracle E-Business Suite, a formal database security program is required.

Evaluating Your Database Security: Are You at Risk?

This white paper is written for senior IT and executive management, as well as stakeholders in legal compliance and risk assessment. The purpose of the paper is to evaluate the effectiveness of your organization’s database security program.

Database Security Responsibility and Vendor Security Patches

The responsibility of securing a database starts and ends with the organization using the database. Organizations own their data, and legal mandates exist for organizations to protect many types of data. Applications are often the source of many database security vulnerabilities, and while much can be done within a database to promote effective security, databases cannot secure themselves. Simply speaking, security is not provided by the database itself, nor is security provided by the application using the database. Organizations can find it difficult to determine what is needed to secure a database and who within the organization is responsible.

For example, how important are vendor-provided security patches? The truth is, vendor-provided security patches provide little security and are a woefully inadequate security approach. In fact, for many organizations they cause more problems than they help and therefore are never applied.

Database security is not the sole responsibility of database administrators (DBAs). Several key disciplines and components must be in place to secure a database. Database security involves far more than applying security patches or using complex passwords and changing them every now and then. The security of an organization’s data is the responsibility of the entire organization, from executive management to operations and IT, from network and security to compliance, legal, and risk personnel.

Limitations of Traditional Checklist-Based Approaches to Database Security

Effective database security requires a formal program to be in place that addresses people, processes, and tools — the type of program that is described in detail in this white paper. However, database security traditionally has been taught to consist simply of checklists of security best practices recommendations (primarily configuration settings). These traditional tools are excellent starting points, but the regular audits and assessments required to enforce them are costly. The checklists are also incomplete and myopic, often lacking key operational controls. Database security checklists typically mandate an artificial and difficult-to-maintain database configuration rather than focusing on the secure installation, configuration, and operation of databases within the organization. Moreover, these checklists fail because too many exceptions are required to handle applications — especially large applications such as Oracle E-Business Suite, SAP, and PeopleSoft.
The Three Strategic Objectives for an Effective Database Security Program

An effective database security program that goes beyond a fragmented checklist-based approach requires three strategic objectives:

1. Minimize the attack surface.
2. Classify databases and act appropriately.
3. Perform intelligent and business-focused auditing and monitoring.

To meet the three strategic objectives for a database security program, the required processes can be grouped into seven components. All seven components are required, and the order in which the components must be implemented reflects their relative importance.

The Seven Essential Components of an Effective Database Security Program

The seven components required for true defensive depth in a database security program will be discussed at length in this white paper.
Part 1: Database Security

To secure a database, especially large databases supporting ERP platforms such as SAP, PeopleSoft, or Oracle E-Business Suite, there are two primary challenges: complexity and time.

**Complexity Weakens Security**

Complexity weakens security and impedes decision making because the larger and more complex a database or application becomes, the harder it becomes (i) to make good decisions, and (ii) to understand the direct, indirect, and possible unintended consequences of decisions.

**Complex Interoperability Issues**

ERP platforms are designed to support complex business solutions on a global scale, inclusive of processes for finance, human resources, manufacturing, and sales/customer relationship management. A platform’s total technology stack includes the ERP application’s supporting operating system, database, and third-party tools and utilities. Simply ensuring that the various release levels of all these components interoperate successfully together in a robustly secure framework can be a challenge, as the combination of ERP functionality and the supporting technology stack creates a seemingly infinite number of permutations and combinations.

**Complex Database Use and Access Issues**

As shown in Figure 1, modern databases are robust put to complex uses and are touched by many different processes and people. This inherent complexity raises numerous security issues.

![Figure 1: Database security is a complex problem.](image-url)
Time Weakens Security

The older the database, the harder it is to secure. Over time, technical security exploits are found and commonly, the older the database version, the more vulnerabilities exist.

Vendor-Driven Software Obsolescence

One reason why older databases are often less secure is the software product lifecycle. Every database and application software product has its own lifecycle. For example, Oracle Corporation commonly supports software for eight years after general availability, and then support expires as shown in the chart below:

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>GA DATE</th>
<th>PREMIER SUPPORT ENDS</th>
<th>EXTENDED SUPPORT ENDS</th>
<th>SUSTAINING SUPPORT ENDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.7</td>
<td>Sep 2000</td>
<td>Dec 2004</td>
<td>Dec 2006</td>
<td>Indefinite</td>
</tr>
<tr>
<td>10.1</td>
<td>Jan 2004</td>
<td>Jan 2009</td>
<td>Jan 2012</td>
<td>Indefinite</td>
</tr>
<tr>
<td>10.2</td>
<td>Jul 2005</td>
<td>Jul 2010</td>
<td>Jul 2013</td>
<td>Indefinite</td>
</tr>
<tr>
<td>11.2</td>
<td>Sep 2009</td>
<td>Jan 2015</td>
<td>Dec 2020</td>
<td>Indefinite</td>
</tr>
</tbody>
</table>

Table 1: Oracle Database Releases.

Because databases and applications are on different obsolescence cycles, and because only specific tested and certified versions of applications and databases can work together, clients may not be able to address vulnerabilities promptly and may increasingly fall behind.

Falling behind on security patches because of vendor obsolescence cycles is a risk that will be discussed in greater detail. Organizations own their data and becoming beholden to the vendor’s lifecycle can have a detrimental impact on security.

---

Vendors Don't Supply Patches for Obsolete Software

When vulnerabilities are found, database vendors will usually patch exploits only to the latest supported versions of their software. New vulnerabilities may be ignored in older, or what have been deemed obsolete, versions (looking at the chart above, Oracle database versions 8 through 11.1 are in sustaining support, which means they no longer receive CPUs [Critical Patch Updates]). Because of the requirement to use only certified combinations of application and database versions, database security patches or configuration changes may at times need to wait for application patches or configuration changes, if not also upgrades, to first be applied, further adding to the complexity.

Scheduling Downtime for Patch Updates Creates Risk and Is Often Avoided

Complicating the overall situation is the need to schedule business downtime to test changes. Downtime interrupts operations and can have an adverse material impact on company revenue. Databases cannot be tested independently of the application they are supporting. Any patch or configuration change, either for the database or the application, requires a proportionate amount of testing by the business dictated by the size/scope of the patch or configuration change. If the business perceives little-to-no benefit to testing and scheduling downtime to apply security patches, over time security vulnerabilities can easily accumulate.

Knowledge About Your Database Implementation Degrades Over Time

Time also degrades security due to knowledge being lost and decisions being made based on imperfect or incorrect facts. Documentation is lost or misplaced as staff moves, relocates, leaves and/or changes roles. Companies and divisions are also bought and sold. For organizations that have been running Oracle applications for 10–15 years or more, theirs is highly unlikely to be a complete, up-to-date implementation, and it’s unlikely that the staff that first implemented the application will still be supporting it.

Figure 2: Database security decays over time due to complexity, usage, application changes, upgrades, published security exploits, etc.
Database Security Risks Accumulate Over Time

Over time, customizations, interfaces, tools, utilities, and database accounts created to support or manage the application and/or the supporting database tend to accumulate. Such artifacts accumulate and are forgotten about — and create ever-growing security risks, especially when documentation has also been lost and staff is no longer present.

Understanding the Need for a Formal Database Security Program

While database security tends to deteriorate over time, what is needed to secure databases does not: Effective database security requires a formal program to be in place that addresses people, processes, and tools.

Such a program must address application as well as database issues. Applications are often the source of many database security vulnerabilities, as most attacks on a database come through the corresponding application. Database security is not provided by the application; and while much can be done within a database to promote effective security, databases cannot secure themselves.

A database security program is a plan of action, not a piece of software. To secure databases, there are specific processes, procedures, and/or steps that people must perform. Some of these processes and/or steps may involve using software tools, but the security improvement created by performing the steps is created only by the actions of people.

The physical, operational, personnel, and human resource aspects of database security are discussed in Part 2 of this white paper. Here, in Part 1, we describe the technical components that must be in place directly in and around a database to establish an effective database security program.

Three Strategic Objectives of a Formal Database Security Program

Traditional Checklist-Based Database Security Recommendations Provide Only Limited Value

Database security traditionally has been taught to consist of checklists of security best practice recommendations (primarily configuration settings). These traditional tools are excellent starting points, but the regular audits and assessments required to enforce them are costly. The checklists are also incomplete and myopic, often lacking key operational controls. Database security checklists typically mandate an artificial and difficult-to-maintain database configuration rather than focusing on the secure installation, configuration, and operation of databases within the organization. Moreover, these checklists fail because too many exceptions are required to handle applications — especially large applications such as Oracle E-Business Suite, SAP, and PeopleSoft.
The Three Strategic Objectives

An effective database security program that goes beyond a fragmentary checklist-based approach requires three strategic objectives to be met:

1. Minimize the attack surface. This is about reducing your total exposure to security vulnerabilities. Nearly all database security vulnerabilities require a valid database session (connection) to exploit; and it’s noteworthy that a valid session is easiest to gain through an application or configured tool. To proactively defend and protect a database requires that vulnerabilities be minimized as much as possible. In security parlance, this is called reducing the attack surface. To minimize the attack surface, both physical and virtual database perimeters must be inventoried, reduced, and secured, and removing sensitive data is a key step. For organizations running large Oracle applications, the attack surface includes not just the application itself, but all the supporting tools, utilities, and third-party applications — many of which require direct database connections.

2. Classify databases and act appropriately. Overall, a risk-based approach is required to provide database security. The risk-based approach must focus effort and resources on specific organizational data in individual databases. Databases and sensitive data must be appropriately classified to define requirements and to make decisions in response to real-time events.

3. Perform intelligent and business-focused auditing and monitoring. Capturing audit data is easy; using it is not. Audit data must be transformed into actionable information. Monitoring must provide the constant vigilance required to support and enforce multiple layers of defense. Baseline configurations must be protected against drift, and trust in operational processes must be verified.

Seven Components of Defense in Depth

To meet the three strategic objectives for a database security program (plan of action), the required processes can be grouped into seven components (see Table 2). All seven components are required, and the order in which the components must be implemented reflects the relative importance of the component. Due to the inherent risks associated with not having certain processes in place, some must be put in place before others. Lastly, the more time that elapses between implementing the seven components and the initial go-live, the more resources it will take to establish an effective database security program.

The seven steps required for true defense in depth in a database security program are described in Table 2 and Figure 3.
Table 2: Database security program components

<table>
<thead>
<tr>
<th>Inventory</th>
<th>Configuration</th>
<th>Access</th>
<th>Auditing</th>
<th>Monitoring</th>
<th>Vulnerability</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>• An inventory of all databases and sensitive data locations. Once you know where your sensitive data is, you can consider removing it from scope, period; removing it via P2PE; or protecting it with the other options presented in this white paper.</td>
<td>• A measurable database security standard and baseline</td>
<td>• Database access management policies, procedures, and tools</td>
<td>• Database auditing requirements, processes, and definitions</td>
<td>• Database real-time security monitoring and intrusion detection</td>
<td>• Vulnerability assessment and management for databases</td>
<td>• Sensitive data protection strategy – encryption, data masking, redaction, scrambling, tokenization, P2PE.</td>
</tr>
<tr>
<td>• Methods and processes to maintain the inventories</td>
<td>• Periodic validation with compliance to the standard</td>
<td>• Database access profiling and monitoring</td>
<td>• Centralized auditing retention and reporting solution</td>
<td>• Database monitoring definition and tools</td>
<td>• Vulnerability remediation strategy and processes</td>
<td>• Data protection policies, procedures, and tools</td>
</tr>
</tbody>
</table>

Figure 3: Database security program components

- Review existing database inventories
- Define scope of database discovery
- Perform hybrid database discovery
- Identify locations of sensitive data
- Review existing database configuration standards
- Define database security standards
- Implement database access management definition
- Select and implement access solutions or policies for privileged and end-user accounts
- Define auditing for key applications and databases based on compliance and data
- Development auditing requirements for DAM
- Define baseline auditing for all databases
- Define auditing for key applications and databases based on compliance and data
- Implement periodic scanning
- Database inventory
- Database security and compliance requirements
- Database access management
- Policies for database security standards
- Database monitoring and alerting definition
- Log monitoring integration
- Rules for measuring compliance with database security standards
- Data protection requirements with policies
- Data protection implementation process
- Define data protection requirements
- Select and implement data protection solutions for encryption, data masking, redaction
- Develop ongoing encryption implementation
- Implement tokenization and P2PE as necessary
Database Security Program Component #1: Inventory

The first step in creating a database security program is to inventory the in-scope databases. This is not a one-time process. A process is required to scan regularly and identify databases throughout the entire organization. Rogue databases, or unknown databases, when found, must be officially vetted and brought under formal management processes (change control).

Especially for large Oracle ERP applications, production databases are commonly copied or cloned to create test, support, and development environments (see Figure 4). This process is complicated when business-sponsored operational projects request additional database copies of varying lifetimes. As projects reach different milestones, database copies are deleted, and new copies are made. Likewise, new production databases can be introduced.

As new databases are identified, each database must be classified according to what type of sensitive data it contains. This is not a one-time event; ongoing process is needed to maintain information about sensitive data, because just as rogue databases can be found, so too can rogue sensitive data. For example, database administrators commonly create copies of tables before running data fix scripts. The accumulation of such tables, if they contain sensitive data (for example, U.S. Social Security numbers), creates security risks and issues.

Depending on the nature and risk classification of the data, organizations subject to the Payment Card Industry (PCI) data security standard or the Health Insurance Portability and Accountability Act (HIPAA) will need to closely consult their compliance requirements for automated discovery and scanning.

The inventory process to identify both databases and sensitive data must be programmatic. For organizations with hundreds, if not thousands, of databases, tools such as Imperva, IBM Guardium, and Integrigy AppSentry can greatly assist in automating the discovery process.
Key Inventory Deliverables
- Database inventory
- Data inventory for key databases

Database Security Program Component #2: Configuration

Once a database inventory has been created, each identified database must have its configurations hardened according to security best practices.

Configuration management is a core concept of the IT Infrastructure Library (ITIL) change management process. Change management aims to control the lifecycle of assets to enable beneficial changes to be made with a minimum of disruption to IT services.²

Over time, adjustments to database configurations and functionality will be required, and the older the database, the more changes will have occurred. Regardless of the type of change, configuration management ensures that these changes are tested and approved before being migrated to production, and that documentation on when, who, how, and where the changes were made is produced as well.

Before deployment to production, each database (and application) should have its configurations documented, especially for those configurations that define or influence security. Also, as part of the go-live process, each database (and application) should utilize an appropriate secure baseline configuration representing recommended security best practices as defined by leading experts.

Default configurations are rarely, if ever, designed with security best practices in mind. The default configurations for databases are designed for the speed and efficiency of the initial installation and setup. This includes the use of well-known default passwords, simple password requirements, and the creation of high-privileged as well as demo and test accounts (users). Such configurations are vulnerabilities that must be closed.

Configuration Best Practices: Frameworks Create Security through Consistency

Hardening guides exist to secure databases according to best practice security recommendations. The U.S. Department of Defense offers military grade hardening guides for free, including hardening instructions for Oracle databases. These guides are referred to as Security Technical Implementation Guides (STIGs) and while military-grade hardening may not be required for most organizations, the STIG for Oracle represents leading thinking and best practices for Oracle database security.

A best practice recommendation is to use a framework with a layered approach for configuration (see Figure 5), starting with a minimum configuration baseline for all databases, regardless of platform (for example, Oracle, DB2, MS SQL Server, and so on). Ideally, the framework should be heavily influenced by or based on standards such as the Defense Information Systems Agency (DISA) STIG. By using a common framework, a core set of configuration concepts (for example, password requirements) can be consistently applied to all databases where needed and can be extended by platform. By defining a common baseline, compliance controls for HIPAA, Sarbanes-Oxley (SOX), and PCI can then be mapped to a common matrix instead of attempting to maintain separate controls for each independent standard.

Once a baseline has been drafted and applied, regular, if not constant, monitoring is required to ensure configurations do not drift toward insecurity. Guarding against configuration drift not only assists in providing security but also helps ensure overall database and application stability and performance.

Manual configuration management reporting for databases is neither recommended nor usually feasible. Oracle databases offer many possible configuration combinations, and tools such as Integrigy AppSentry are designed to scan and report on Oracle database configurations, including running configuration comparisons against the DISA STIG for Oracle. By using automated tools, the efficiency and effectiveness of database configuration scanning is greatly enhanced.

Another best practice is to scan each database's configurations after each maintenance cycle, before being released back into a production state. This way, only those databases using a secure and safe configuration are allowed back into production. Tools such as Integrigy AppSentry should be deployed to automate such database configuration scanning. Operating systems can likewise have their configurations monitored. Tools such as Tripwire and OSSEC provide solutions for file integrity and configuration monitoring.

Key Configuration Deliverables
- Database security and compliance requirements
- Database security standards

Database Security Program Component #3: Access Management

Once the inventory has been created, and the database hardened per recommended security best practice configurations, access management must be reviewed. Almost all database security vulnerabilities require a valid database session (connection). Ensuring that both existing database accounts are valid, and that accounts created in the future will also be valid, requires several key processes to be in place.

Reconcile and Cull Accounts
The first step in access management is typically a deep audit of each database's users and accounts, followed by a cleanup effort. If users cannot log on (connect), the number of vulnerabilities they can exploit are dramatically reduced, and in most cases eliminated. Stale, unused, and obsolete accounts must be dropped. Over-privileged accounts must be reduced.

Least Privilege and Drift
The concept of least-privilege per one's job function should be the guiding principle when creating and auditing accounts. Granting full control over the database to an account for no other reason than the requestor did not know what privileges he or she needed is a security violation. The most important privileges to secure within a database are those that define and govern the database. Users with these privileges have full control over the entire database, including the ability to read and to change any and all data within it. Those users are commonly referred to as DBAs, and DBA privileges should only be given to high-trust staff and/or employees.

Granting DBA privileges unnecessarily, especially when granted to accounts with well-known default passwords, will easily defeat any and all security measures, both within the database and the supporting application. Finding database accounts with DBA privileges (or significantly elevated privileges) is common. Such occurrences are almost always not malicious, but rather the result of oversights, poor design or support, and/or development activity.

Large organizations and large applications face significant risks with access management over time, as applications are updated and changed, and accounts are also changed due to human resource activities (hires, transfers, layoffs, and so on). Consequently, if not addressed, least-privilege security deteriorates as privileges accumulate and/or are forgotten.
Regular audits of accounts and privileges are required to enforce least-privilege appropriate to job function. These audits must also largely be automated. Tools such as Integrigy AppSentry, Imperva, and IBM Guardium can assist with these audits and report on changes over time.

Key Access Management Processes

- **Provisioning** — The process of creating database user accounts and determining what privileges are granted to the accounts is referred to as provisioning. Provisioning must be a formal process that is thoroughly audited and monitored. Without formal provisioning processes, database access management will quickly disintegrate into weak and ineffective security controls. All accounts must be documented, even those created and used by applications. It is strongly suggested to use the classification scheme depicted in Figure 6 below to systematically classify accounts by usage and associated security risks. At a minimum, accounts should be grouped into three primary categories: database, application, and user.

- **Authentication and Authorization** — Who can log in to the database and what they can do once they log in is referred to respectively as authentication and authorization. Authentication defines whom you trust to log in to your databases and authorization defines what you trust them to do once they have logged in. Database authorization is very complex. The number of features provided by databases such as Oracle allows a very large number of privileges to be granted. Privileges include who can create users, read data on certain tables, or read data from any table as well as who can update and change data.

- **Administration** — Who is allowed to edit or alter accounts and when and why accounts are edited must be formal processes. Each change to an account must be documented; otherwise the change may be deemed a security event.

- **De-provisioning** — All accounts follow a lifecycle. Accounts that are no longer used create multiple security vulnerabilities and must be closed. Like provisioning, a formal process is required to document who, when, how, and why each account is closed.
Access Management Must Be Automated

As much as possible, authentication and authorization should be integrated with the overall corporate identity management approach and strategy. Having a clean identity management source of truth will strengthen security by allowing cleaner designs and requirements with fewer exceptions. Ideally, named database accounts for staff and employees should be authenticated using the corporate Lightweight Directory Access Protocol (LDAP) or Active Directory. LDAP and AD group membership should also drive authorization decisions.

For building on an identity management source of truth for an organization, database security tools such as BeyondTrust’s PowerBroker and password safes such as CyberArk are invaluable. Password safes can physically control access by restricting passwords to least-privilege or need-to-know. Without a password, a database connection cannot be made. By basing access to passwords directly on LDAP or AD group membership, passwords can be safeguarded and key access management processes can be automatically and programmatically enforced.

For example, with a password safe such as CyberArk, Oracle database passwords can be separated from SQL Server and/or network assets and operating system root accounts.

Tools such as PowerBroker can take this concept further, adding restricting privileges, as well as passwords.

Access Management Key Deliverables
- Database access management
- Policies for database account management
Database Security Program Component #4: Auditing

Once a database has been inventoried, its configurations hardened and subjected to effective access management practices, much of the effort to secure it is complete. To maintain an effective security posture, and to verify the trust of privileged users such as DBAs administering the database, auditing must be introduced.

Auditing provides the data with which to verify trust. Users are authorized only for certain privileges. Auditing identifies when users’ privileges have changed or when they exercise privileges that they potentially have not been authorized to have and/or use.

A common audit framework should be applied to all databases. This will give a single lens through which to view database activity and make security decisions. The foundation of the framework should be a set of security events and actions that are audited and logged in all databases (see Figure 7). These security events and actions should be derived from and mapped back to key compliance and security standards most organizations have in place to comply with such as HIPAA, PCI, and SOX (see Figure 8).

Database auditing cannot be done manually, and audit logs must be sent to a centralized solution for safekeeping, segregation of duties, and monitoring (correlation and reporting). Such tools are referred to as Database Activity Monitoring (DAM) solutions.

DAMs work through the use of agents deployed on database servers, and/or by intercepting and reading network traffic between clients and servers. Agents and intercepted traffic are relayed to a centralized DAM solution for analysis. One advantage of DAM solutions is that they can be implemented transparently in databases and applications. No configuration of native database auditing is required for DAM solutions.

Leading DAM solutions include Imperva and IBM Guardium.

Key Auditing Deliverables
- Database auditing definition for (1) all databases and (2) key databases
Figure 7: Framework of security events for database auditing

Figure 8: Mapping framework for database auditing
Database Security Program Component #5: Monitoring

Capturing audit data is easy; using it is not. Audit data must be transformed into actionable information to make decisions about violations of trust. The process of making such decisions is referred to as monitoring. Monitoring provides the constant vigilance required to support and enforce multiple layers of defense. Monitoring allows baseline configurations to be protected against drift and trust in operational processes to be verified.

A database security program must have a monitoring solution that is outside the database. Not only does this create a segregation of duties by placing monitoring outside the reach of DBAs, but it also allows correlation among and between databases and other assets (such as firewalls, VPN activity, and applications).

The overall success of a database security program is relative to the investment made by organizations in monitoring (see Figure 9). The design and implementation of auditing lends itself to a one-time project, whereas monitoring must be supported 24x7 for every day after that. To have an effective database security program, organizations must have an appropriate number of staff to support monitoring according to the size and number of databases being monitored.

Monitoring is constant and persistent; it must be performed by an automated solution and cannot be done manually, especially for large active databases and for large numbers of databases. Automated DAM solutions such as Imperva and IBM Guardium create the audit logs. Automated monitoring solutions such as Splunk and ArcSight provide correlation, reporting, and alerting.

Figure 9: Framework for database security program
Decision-Making Runbooks

The success of monitoring relies on the quality of the runbooks. Runbooks are the response matrices defined for each event being monitored. Runbooks for each event provide formal instructions for what decisions and actions must be performed by whom and when.

An example of a runbook is when a database user is created and then immediately granted full DBA privileges. Such events should be rare and infrequent, and due to the potential security risks, easily warrant a confirmation. When detected by auditing, a severity one (highest priority) alert should be raised. The runbook for this event would include the phone numbers and personnel to contact and the decision required.

In this scenario, the runbook would identify that both the DBA and IT security teams should be immediately notified — sending an email and opening a ticket will not suffice. Assuming a 24x7 security operations center or service is raising the alert, phone calls would be made until a “live” person was reached (no voice mails). The runbook would specify to find a “live” person by calling the defined list and then escalate up the chain-of-command until a decision maker was reached — possibly as far as the CIO or CEO. The decision makers would then need to determine if the newly created account is malicious or whether an approved ticket or service request exists. Two decision makers are needed to ensure segregation of duties is enforced and to promote better decision making (two people will usually allow a better decision to be made than just one).

Without runbooks, information generated by monitoring becomes stale, and opportunities to take action are lost (for example, closing compromised accounts in a timely fashion before sensitive data is lost). Runbooks must exist in detail proportionate to the risk defined by the data.

Key Monitoring Deliverables
- Database monitoring and alerting definition
- Log monitoring integration

Database Security Program Component #6: Vulnerability

Once the auditing and monitoring components are in place, vulnerability management processes can be introduced. One reason for this is the principle of triage. When securing a database, vulnerability management can be only truly effective once the database has been hardened and then, through auditing and monitoring, a means is in place to maintain the security posture.

Vulnerability management is the process that identifies and guides decision making about risk. The risk is always defined by the data, and the vulnerability management process is sometimes erroneously focused on vendor security patches. Organizations own their data; organizations need to secure their data.
Where Do Vulnerabilities Come From and Why Do They Exist?

Vulnerabilities can take the form of security bugs with vendor-written code, or deviations (drifting) from best practice-based configuration settings. Since databases exist as part of a solution that includes the supporting operating system in addition to the application that the database supports, applying database security patches and making database configuration changes is rarely timely, and in some cases it is not possible at all.

Perfect bug-free software does not exist. There may be exceptions, hopefully, such as the military’s command-and-control system for nuclear weapons, but overall because humans write software, security vulnerabilities will always exist due to design flaws and stupidity, and because of the inherent cleverness of the human species.

Vulnerabilities also exist because it is profitable to find them. Security researchers seek out and commoditize vulnerabilities to market their services and/or products. Likewise, well-funded malicious organizations such as the Russian mob, Nation State actors (for example APT1) and political hacktivists (for example, Anonymous) seek out vulnerabilities to exploit for both financial as well as political gain.

Do Not Depend on Software Vendors for Security!

Vulnerability management is a far larger topic than configuration management, and the two should not be confused. True vulnerability management involves much more than looking for insecure configurations, worrying about zero-days, and, in general, running on the vendor security patch “hamster-wheel” where great amounts of resources can be expended and relatively little additional security provided.

Relying on vendor security patches and vulnerability processes does not work for several reasons:

- Many vendor patches do not work; they do not fully address the vulnerability and/or some patches cause other problems.
- Sentrigo Inc., a database security products vendor conducted a survey that found a full two thirds of Oracle DBAs (206 out of 305) were noted as never having applied Oracle CPUs.
  - There are two major reasons for the trend, Slavik Markovich, chief technology officer at Sentrigo, said. The first and most important is that most DBAs fear the consequences of installing a patch on a running database, he said.
  - “To apply the CPU, you need to change the binaries of the database,” he said. “You change the database behavior in some ways that may affect application performance,” he said. So applying security patches to a database typically involves testing them against the applications that feed off the database, he said. “This is a very long and very hard process to do, especially if you are in enterprises with a large number of databases and applications,” he said. Applying these patches means months of labor and sometimes significant downtime, both of which most companies can’t afford, he said.
Some application vendors also don’t certify Oracle patches to run with their applications, Markovich said. As a result, a database administrator might, for instance, be wary of installing a patch on an Oracle database that is being used by a SAP application because that might be grounds for the application vendor to refuse addressing any disruptions to the application, he said.

Another problem is that companies that want to install the most recent Oracle patches need to first ensure that they have already installed the previous patch set, Markovich said. So companies that fail to keep up with the latest patches keep falling further behind with each patch set release, he said.

Other vendor patches are only released for specific modules or products, not all.

Still other vulnerabilities are addressed only for those customers paying for premier or enhanced support.

It should also always be assumed that there are an unknown number of vulnerabilities yet to be found and/or disclosed. Some of these will never be patched or patched properly by the vendor.

Lastly, vendors exist to make profits, and maximizing security reduces profits. Relying exclusively on a vendor’s vulnerability management processes and decision making will not secure an organization’s data.

When vendor database security patches are released, making changes to the database might need to wait until the application’s vendor certifies the new permutation of combined patch levels. Each change to a database must be made according to the application’s overall tested combination of patches and configurations for the operating system, database, and application as well as all supporting utilities and third-party tools that comprise the overall application.

Likewise, large applications and organizations usually require longer testing cycles, and it is often difficult to schedule system downtimes with operations. The entire process of testing and applying patches and making configuration changes to databases to mitigate vulnerabilities can take weeks, if not months. In this context, applying security patches can lead to overall insecurity if resources are diluted in pursuit of mitigating low-risk vulnerabilities with little security impact.

Process for Decision Making

Vulnerability management is the formal process for identifying and guiding decision making about risk. It includes the day-to-day project management of mitigating risks, including working with the business to test and negotiate downtime to make changes. It also includes identifying workarounds, both temporary and, at times, permanent.

Vulnerability management must address the risks as a whole and starts with the data; sensitive data and databases must be inventoried. The risk calculation consists of what data is specifically at risk, the technical exploit required, and the likelihood and effort required. For example, sensitive information, if encrypted, will be protected when at-rest. If a vulnerability exists where data at-rest is an issue, the risk may be deemed lower if not mitigated.

Automated tools must be used to scan databases to identify vendor-reported database vulnerabilities, and to maintain secure configuration baselines. These tools should be integrated into an organization’s Security Information and Event Management (SIEM) to assist in giving management overall dashboard views into vulnerability risks. Imperva, IBM Guardium, and Integrigy AppSentry all provide such functionality. To attempt these tasks manually is labor-intensive and not feasible.

As part of a vulnerability management program, the scan results should be reported to senior management on a regular basis, ideally several times a year. The results of the scans must be analyzed and quantified. The analysis must include configuration baseline drift and strength of access management, auditing, and monitoring processes.

A critical component of vulnerability management is communication, especially with senior management. Mitigating high-risk vulnerabilities will require strong executive management sponsorship.

Key Vulnerability Deliverables
- Rules for measuring compliance with database security standards

Database Security Program Component #7: Protection

Banks are robbed because that’s where the money is. It’s similar with databases. Databases are hacked and compromised because organizations store their sensitive data in databases. While it may seem counter-intuitive, the protection of sensitive data is the last of the seven components to be put in place. It does not make sense to put effort into securing sensitive data in the database until the database as whole is secured — in other words, since banks are robbed because that is where the money is, it does not make sense to put money in the bank until the bank has a safe and the safe can be locked.

Securing sensitive data requires a formal process for several reasons. Securing sensitive data cannot be left to guesswork and optimistic assumption. Usually there are multiple legal and compliance requirements and mandates to protect sensitive data. Another reason is that sensitive data is hard to find, and once located, is difficult to keep track of.

Securing sensitive data is also complicated by the restrictions posed by the features and functionalities of both the application and the database itself. Not all technical solutions will be feasible, and the exact technical solution deployed will be highly dependent on the technology stack.

Of all of the seven components of database security discussed in this white paper, protection is the hardest to implement.

Sensitive Data Protection

Point-to-Point Encryption (P2PE) and tokenization have gained tremendous traction in the payment processing arena in order to comply with PCI. Fully complying with PCI is expensive and time-consuming, and doesn’t necessarily mean you will be protected from a breach. Several of the high-profile breaches in the last few years come from companies that successfully passed third-party PCI audits. The best way
to protect your organization from a breach is to outsource payment processing via P2PE and replace cardholder data via tokenization. This cost-effective risk mitigation option, with prices under $.01 per transaction depending on volume, should be a top priority for anyone still storing payment data, regardless of whether they have access to vendor patches.

Because of the success of P2PE and tokenization for payment data, the industry has begun to look to other areas where sensitive data can be moved to more secure locations. Protected Health Information (PHI), covered by HIPAA, is fast becoming an option for P2PE and tokenization.

A process for the protection of sensitive data is depicted in Figure 10. To protect sensitive data, a risk-based approach should be used to direct appropriate resources to the highest-risk data and databases. The process starts with the privacy policies of the organization. These policies determine what data is sensitive, who can access it, and acceptable solutions for protecting it. Client contractual, regulatory, and compliance requirements usually heavily influence organizational privacy policies.

Using the privacy policies, the database inventory and discovery processes then identify the sensitive data specifically to be secured. The inventory of sensitive data usually is kept in the following format: server-database-table-column-steward.

The steward is the employee responsible for the security of the specific data element. Typically, the data steward is the business owner. For example, the senior human resource officer will be the data steward for U.S. Social Security numbers — in large part because of the senior human resource officer’s fiduciary responsibilities as defined by HIPAA.

Figure 10: Data protection process
The concept of “live data” is important to understand when deciding how to secure sensitive data. Data is often copied, as noted earlier, from production to test and development environments as well as to data warehouses. Regardless of where data exists, if it is live-production sensitive data, it must be protected. Constant vigilance for the detection of rogue sensitive data is required. Copies and backups of tables and sometimes of entire databases are often much softer targets to hack and compromise. Rogue data and databases are previously unknown and/or not inventoried.

Automated tools should be used for the sensitive data inventory and scanning process. These tools can also be used for configuration management and should be integrated into your SIEM when rogue sensitive data is found. Imperva and IBM Guardium provide such functionality. These tools use technology such as regular expressions to parse each packet of data coming from a database to determine if certain patterns, such as credit cards or U.S. Social Security numbers exist. Rules can then be created and applied and usage can be mapped and audited. Additionally, rules can be defined to dynamically limit or block the use of sensitive data.

Once located, specific data protection solutions must be deployed to secure the sensitive data. Depending on the database’s production status, different tools and approaches will be used to secure the sensitive data within it — including tokenization and encryption solutions such as those offered by CardConnect, Vormetric, and Voltage. The regulatory and compliance requirements for the protection of sensitive data determine, to a large extent, what tools and approaches should be used, as well as the attestation (such as logs and reports) required to prove that the protection is in place and working.

The selection and implementation of sensitive data protection solutions will also be highly dependent on the functionality restrictions of the database and application. Some applications may prohibit truncating records or tables as well as the scrambling of records. Afterwards, testing is always needed to prove that applications still function. The solutions must then be documented and staff training may also be required.

Vendors such as Vormetric and Voltage both provide solutions for the protection of sensitive data. Table 3 provides a high-level description for the approaches used to protect sensitive data.
Encryption

Encryption is often discussed as a solution for the protection of sensitive data. Some organizations may even be forced to encrypt data because of client contractual, legal, or compliance mandates. Encryption, while useful, has several drawbacks.

Aside from certification and integration requirements with applications they support, with database encryption there usually is a statistically measurable performance impact. Where encryption is supported, there is a performance cost; and where encryption is supported, usually more is not better.

Encryption is not an access control solution. All encrypted data in a database will be encrypted the same, regardless of the user creating or viewing the data. Database encryption is very similar to a hotel where all guests get the same room key. Non-hotel guests cannot get into the hotel, but all guests have full access to all guest rooms. Encryption of sensitive data does not protect against malicious privileged employees and contractors. For example, DBAs and developers with direct access to the database will in many or most cases have full access to unencrypted data.

Encryption is a coarse-grained security solution, and, where feasible, is usually a good addition to the defense in depth for data-at-rest outside the database; it is excellent for protecting database backup files and helping to mitigate the risk of database data files being compromised either through hacking activity at the operating system level or through the theft or loss of storage media.

Key Protection Deliverables

- Data protection requirements with policies
- Data protection implementation process

Table 3: The approaches used to protect sensitive data

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purging and Truncating</td>
<td>Removing sensitive data is usually the safest way to secure it. If the sensitive data does not exist, it cannot be compromised. Application design often dictates whether this can be done.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Encryption is at best a coarse-grained access control tool. Databases and applications usually deploy encryption at the file system level to be transparent to the database and/or the application. This means that once the database is in use, it is decrypted. It also means that DBAs and developers also have access. Encryption, other than meeting compliance mandates, usually helps mitigate only the security risks of the theft of servers and/or storage arrays and backup files.</td>
</tr>
<tr>
<td>Masking or Scrambling</td>
<td>Masking or scrambling of data is the replacing of sensitive data. The replaced data can either be randomized or use an algorithm. Often non-production databases employ the technique of scrambling to protect sensitive data before releasing test and development databases to testers and developers.</td>
</tr>
<tr>
<td>Redaction</td>
<td>Often confused with masking, redaction is the filtering of results. The filters are security rules that determine what people are trusted to see what sensitive data. For example, see the full social security number, see only the last four, or see only “#” characters. Oracle Advanced Security allows for real-time redaction, which is transparent to applications.</td>
</tr>
</tbody>
</table>
Interdependencies and How to Start

The seven components of the database security program must be implemented sequentially. Each component builds on the foundation of the prior components. The order of implementation reflects the relative importance of the component and the inherent risks associated with not having certain processes in place before others. Lastly, the more time that elapses between implementing the seven components and the initial go-live, the more effort it will take to establish an effective database security program.

Typically, database security programs are put in place in three phases:

- Planning
- Implementation
- Ongoing

Planning

Building a database security program requires planning if for no other reason than that most organizations cannot implement all of the components at once.

To effectively size and scale the components, the inventory of in-scope databases and the sensitive data discovery within these databases must occur first. Once the inventories have been completed, planning can commence on data protection and the section and implementation of a DAM solution. This selection process may involve a pilot or proof-of-concept before making an investment.

Implementation

The implementation phase usually centers on the implementation of the DAM and sensitive data protection solutions. Access management also commonly involves large cleanup efforts in the implementation phase as rogue accounts are purged and consistent standards for authentication and authorization are introduced.
Ongoing

Because databases are constantly being changed as new data and configurations are created, security must be an ongoing effort. All of the seven components have annual, if not quarterly or even daily, processes that must be sufficiently staffed and correctly executed.

Over time, databases change size and new databases are created. Rogue databases and data must be purged. Configurations and setups drift and must be corrected. All of this requires constant vigilance.

Figure 11: Database security program implementation
Part 2: People and Physical Security

Part 1 of this white paper describes seven technical security components that must be in place directly in and around a database. Part 2 discusses the physical and operational, personnel, and human resource measures that are required to secure a database.

Databases do not exist by themselves. Databases exist to store information created by applications, and applications are used by people. People also support and maintain applications and databases, as well as the entire technical environment and supporting infrastructure for a database.

Figure 12 summarizes the relationship of data security to people and physical security. Concentric perimeters of security protect data. The innermost layer is the database itself and how it is configured. Outside of the database are the physical and infrastructure security features that protect access to the database. The outermost layer of security is people. Decisions must be made about who should have what access, how, and when.

Figure 12: The relationship of data security to people and physical security

People Security Perimeter

Database security, as with any security topic, begins and ends with people. People’s decisions and actions lead to security events and insecurity. In terms of database security, people who have privileged access to use, maintain, and administer the database, or who have elevated privileges to all data within the database, are of paramount concern.

The scope of this white paper does not allow a detailed discussion of operational security processes. At a high level, who you should trust to be privileged database users and why you should trust them should be part of a mature human resource program that includes the use of background checks. Trust must be verified, and background checks are a vital tool to identify what people should be trusted with what data and how much they should be trusted.
Background Checks
If your database security program does not use background checks, or uses cheap and ineffective $50 background checks, or does background checks only once at hire, you are not secure. Privileged database users must undergo extensive and thorough background checks. If you outsource, you need to receive attestation from your vendors that background checks are being performed, in the same way you would investigate the background of one of your employees.

Outsourced Employees
Another common oversight with people security is that managed services, cloud providers, and hosting companies commonly use contractors and third parties. Offshore operations centers are often separate legal entities, if not completely separate third-party companies servicing multiple clients from the same location. You own your data. You need to ensure that your security requirements flow down to each person who has access to your data.

The Security Department and Its Mission
Lastly, concerning people security, do you have a security team or department? Whom do they report to and what is their mission? Is their focus myopically only on the network, desktops, and antivirus issues, or do they also consider database security? What do they know about database security? Is the team large enough, based on the number of databases? If you do not have a security team looking at database security, you are not secure.

Physical and Infrastructure Security Perimeters
Regardless of whether or not you want to trust people, your trust perimeter is physically defined by who has physical access to the servers (either directly or via a network) and who has the passwords.

Managing Physical Security
Physical security is the truly medieval stuff: locked doors, walls, gates, and armed men. If you do not have effective physical security, you are not secure. Even if your servers are encrypted, if you have physical access to a server, you can more or less assume a hostile actor can compromise it. Physical security is also wholly dependent on people security. Physical security filters access for people who either should or should not be trusted.

If you are effectively restricting physical access to your database servers, databases are rarely accessed from the database server’s console — for example, by accessing the database directly from the server console inside the data center. DBAs are rarely (usually never) allowed access to the data center. Database access is mostly through applications, tools, and utilities.

Work areas for DBAs and high-privileged and trusted employees must be restricted. For example, they should not be on the first floor near windows, and their laptops should be encrypted and have locking cables on them. If a DBA lost their laptop, what would be risked?
Network Design

Network design has a large impact on physical and infrastructure security for databases. How flat is your network? Do you use DMZs? Do you have both internal and external DMZs? Do you have separate network segments for applications, databases, storage, and administration (for example, hypervisor consoles and monitoring)? Or can your databases be directly accessed from any wireless access point or conference room wall jack in your office? Often an organization will implement controls such as firewalls, proxies, intrusion protection, and monitoring and then fail to implement them for internal users. Do you still allow any-any access between non-related servers in the data center? If you are outsourced, are your databases directly accessible from any conference room or wireless access point in your vendor’s locations in throughout the U.S., Asia, and Europe?

Managing Passwords

Another indication of overall IT security maturity is how passwords are managed. Large Oracle applications, through the installation processes, create hundreds of passwords for a single environment (operating system, database, and application). If password safes are not being used to safeguard and protect passwords, and to segregate access to passwords to appropriate personnel only, overall security usually is weak. For example, are Oracle database passwords separated from SQL Server, network assets, and operating system root accounts?

Still another measure of security is use of tools such as Puppet and Chef. Puppet and Chef automate the installation, configuration, and maintenance of servers. These tools greatly promote the use and maintenance of secure baseline configurations for operating systems and virtual machine guests.

Lastly, is your organization ISO 27000 certified, working toward an ISO 27000 certification, or using ISO 27000 as a guiding principle? The International Standards Organization (ISO) 27000 series are best-practice security standards for information security. ISO 27001, in particular, is a formal specification concerning the management of information security risks and is equally applicable to all types of organizations, both public and private.

---

Conclusion

Effective database security requires a formal security program consisting of seven interdependent components:

1. **Inventory**
2. **Configuration**
3. **Access**
4. **Auditing**
5. **Monitoring**
6. **Vulnerability**
7. **Protection**

All seven components are required and are comprised of formal processes, executed by people using tools: security is a process and tools do not provide security, people do. The chief thrust of the seven components can be summarized as follows:

1. **Limit access to the database** — Create physical and virtual perimeters to reduce access, especially direct database access, as much as possible.

2. **Classify databases and act appropriately** — Data must define the acceptable level of risk for each database. Focus resources on the most important data and databases first. Use a layered approach with a minimum configuration baseline as a consistent framework to be applied to all databases.

3. **Verify trust** — Know who to trust and verify their trust. Especially for direct database connections outside applications, auditing must enforce and verify trust and, in general, must transform audit data into actionable, business-focused information.
Appendix A: Tools Cited

The table below summarizes the tools mentioned in this white paper.

<table>
<thead>
<tr>
<th>TOOL</th>
<th>PURPOSE</th>
<th>WEBSITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppSentry</td>
<td>Inventory, Configuration Management, Auditing, Vulnerability</td>
<td><a href="http://www.integrigy.com/products/appsentry">http://www.integrigy.com/products/appsentry</a></td>
</tr>
<tr>
<td>CardConnect</td>
<td>Protection</td>
<td><a href="http://www.cardconnect.com/">http://www.cardconnect.com/</a></td>
</tr>
<tr>
<td>CyberArk</td>
<td>Password Safe, Access Management</td>
<td><a href="http://www.cyberark.com/">http://www.cyberark.com/</a></td>
</tr>
<tr>
<td>Imperva</td>
<td>Inventory, Configuration, Auditing, Monitoring, Vulnerability, Protection</td>
<td><a href="http://www.imperva.com/">http://www.imperva.com/</a></td>
</tr>
<tr>
<td>OSSEC</td>
<td>O/S Configuration Management</td>
<td><a href="http://ossec.github.io/">http://ossec.github.io/</a></td>
</tr>
<tr>
<td>Splunk</td>
<td>Auditing &amp; Monitoring</td>
<td><a href="http://www.splunk.com/">http://www.splunk.com/</a></td>
</tr>
<tr>
<td>Tripwire</td>
<td>O/S Configuration Management</td>
<td><a href="http://www.tripwire.com/">http://www.tripwire.com/</a></td>
</tr>
<tr>
<td>Voltage</td>
<td>Data Protection</td>
<td><a href="https://www.voltage.com/">https://www.voltage.com/</a></td>
</tr>
<tr>
<td>Vormetric</td>
<td>Data Protection</td>
<td><a href="http://www.vormetric.com/">http://www.vormetric.com/</a></td>
</tr>
</tbody>
</table>
About Rimini Street, Inc.

Rimini Street is the global leader in providing independent enterprise software support services. The company has redefined enterprise support services since 2005 with an innovative, award-winning program that enables Oracle and SAP licensees to save up to 90 percent on total support costs. Clients can remain on their current software release without any required upgrades for at least 15 years. Over 1000 global, Fortune 500, midmarket, and public sector organizations from a broad range of industries have selected Rimini Street as their trusted, independent support provider.

Rimini Street and the Rimini Street logo are trademarks of Rimini Street, Inc. All other company and product names may be trademarks of their respective owners.