Secure Coding.
Practical steps to defend your web apps.

Setting Up a Database Security Logging and Monitoring Program

GIAC (GCIA) Gold Certification

Author: Jim Horwath, jim.horwath@rcn.com
Advisor: Rodney Caudle

Accepted October 10, 2012

Abstract

Many corporations concentrate the majority of their resources towards securing the perimeter of their networks, often neglecting the most critical company asset, databases. This results in a security architecture that resembles a hard-shell chocolate candy, crunchy on the outside, gooey and vulnerable on the inside. Databases contain the most valuable data companies own – customer, employee, financial and intellectual property to name a few categories. Protecting, logging, and monitoring database data should be a core activity of every business, unfortunately many businesses fail to provide adequate security logging and monitoring for their databases. This paper will discuss implementing a security and logging program with the end goal of increasing the protection of data.

1. Introduction

This paper is about implementing a database security logging and monitoring program to increase the security posture of a corporate infrastructure. Databases present unique challenges due to the volume and dynamic nature of the content. The data is often the most sensitive and valuable that a corporation posses. Most corporate data deals with sensitive information concerning customers, employees and partners. This sensitive data often falls under federal and industry regulations that incur a monetary or a reputation penalty in the event of a data breach. Regulations typically levy fines associated with violations and loss of reputation can destroy a company. Implementation of a successful database security logging and monitoring program needs a triangle of processes and procedures, compensating controls and education. There is no magic bullet that once implemented will solve and meet all business needs. There needs to be a combination of things in order for the program to be successful, if one piece of the triangle (processes and procedures, compensating controls and education) is missing, the program will be in danger of not meeting its desired goal – securing and monitoring data.

Data and information are the most critical assets a company has. Loosing data exposes a company to litigation and loss of reputation. Information stored in databases is important; otherwise, there would be no reason to store the information. Companies routinely store sensitive, private and proprietary information such as social security numbers, credit cards, payroll records, personal information to name a few. Companies must maintain and secure this information in a confidential manner or expose themselves to loss of reputation and/or revenue.

Attempting to address every exposure and vulnerability is a lofty aspiration that may result in missing the goals set forth. This paper will concentrate on addressing common database security vulnerabilities and threats. Protecting against these common vulnerabilities and threats will help protect an organization's sensitive data. Policy is important for any security program, if a company does not

know what to protect it cannot effectively safeguard it. SQL Server will serve as the example for this paper because of its popularity and wide adoption by businesses. This paper will concentrate on addressing common database security threats and using a combination of a commercial tool, policies and configurations to protect a database from the security threats. By concentrating on the common database vulnerabilities and threats, database security will increase substantially. The commercial tool will address things in a generic manner where the concept can extend to any tool on the market.

The paper would not be complete without a discussion of the various options available for database logging and monitoring. This discussion helps decide which techniques will fit into an environment for solving business needs and issues. There is also a discussion around the use and implementation of database encryption, because many decision makers and vendors see encryption as the panacea for database issues. It is a complicated subject, demanding understanding and thought before implementation. Finally, the paper concludes with a few case studies of common database vulnerability issues and compensating controls implemented to meet business needs. (Ottman, 2010)

2. The Common Database Vulnerabilities List

This paper will focus on addressing a list of top database vulnerabilities as a driver its use cases. The Internet has many database vulnerability lists; the list included in this paper seemed to be a good mixture of common vulnerabilities found during research. A solid database security logging and monitoring project will address the most common database vulnerabilities and use them as business drivers. Solving the common database problems will help protect sensitive data and an organization from the most common vulnerabilities. Once an environment implements controls against the common database vulnerability list, the security posture of the environment will improve substantially. This section will define common database vulnerabilities along with the potential risk they pose to a corporation. Failing to know what needs protection and what to protect them

against is a recipe for failure. The list of common dataset vulnerabilities is in the table below. (Shulman, 2006)

Common Database Security Threats
Excessive Privilege Abuse
Legitimate Privilege Abuse
Privilege Elevation
SQL Injection
Weak Audit Trail
Denial of Service
Database Communication Protocol Vulnerabilities
Weak Authentication
Backup Data Exposure

Common database security threats

2.1. Excessive Privilege Abuse

One of the principles of security is least privilege; a user should have the least amount of privilege necessary to perform their job function or task. Granting a user privileges beyond their need is a common practice and can lead to abuse from excess privilege. For example, an administrative assistant may need privileges to verify vendor and business partner payments. However, if he/she has the privilege to authorize and create payments there is the opportunity for fraud because the assistant has authority beyond their job responsibilities. Many times database users receive a generic group privilege with the purpose of one-size fits all. This results in granting excess privileges to users who do not need them for job functions. Granting the right amount of privilege and being able to detect excessive privileges can be time consuming, but will help increase the security posture of an organization. (Ottman, 2010)

Developers and support staff typically like to have the maximum amount of privilege while doing their work; most times, they want privileges beyond what they need for their duties. This is a common occurrence during the development cycle. Using a privileged account can become an easy fix for development and support staff lacking the understanding of the symbiotic relationship between an application and the database.

Using credentials with a high amount of privilege will often mask configuration issues because the privilege account can access resources without any issues. This mistake will have consequences when the development effort moves into a production environment and the application requires higher privileges than it should to run. It can be difficult to support an application that runs with high privileges because the application many times can perform operations that it was never intended to do and may mask logic or security errors. If there is an exposure that allows the attacker access to the system after the exploit, the attacker has privileged access to the system. If the application runs on a database supporting multiple applications, it can unintentionally have access to data outside of its application scope. Additionally, when users connect to the database with privileged service accounts the audit trail of who did what can become cloudy. This violates best practices, audit and compliance regulations. (Natan, 2005)

2.2. Legitimate Privilege Abuse

The second form of privilege abuse concerns users who require privilege for their job function, but abuse these privileges for unauthorized uses. Job responsibilities of infrastructure administration staff depend upon elevated privileges for their job functions. Likewise, staff responsible for applications with sensitive or privacy information requires special privileges. Staff in positions of authority with privilege can abuse their authority for unauthorized access to data, or perform actions beyond their job duties. An example of this is using privileged or service accounts for routine duties. Using accounts with high privilege for completion of daily tasks introduces additional exposure. This is important with sensitive data access. There is the additional issue of non-repudiation and associating service account usage with an individual. If there is an employee looking at the personal data of high-profile accounts, knowing who is accessing the data, and why is important. If the user is using a privileged account and becomes the victim of malware, the privilege account can quickly spread into sensitive and protected areas passing through controls and protection systems. Monitoring and reporting on abuses of privilege are difficult, but must be an integral part of any database security monitoring and logging program. (Ottman, 2010)

2.3. Privilege Elevation

Another common form of privilege abuse results from a user or process elevating their privileges to increase their authority. Privilege escalation exploits a flaw or bug in software allowing a user to access resources that the user cannot access under normal circumstances. This allows the user to have greater privileges than the developer or application intended. For example, a user of low privilege may find vulnerabilities in a stored procedure that allows them to assume administrative privileges granting them control over the entire database. A database contains many opportunities for a malicious user or process to elevate their privileges through database stored procedures and SQL statements. Once a malicious user has administrative privileges they own the database and its data. This form of privilege abuse has the potential to be devastating to a corporation. Security researchers routinely publish flaws and exploits; it is very easy for a malicious user to find vulnerabilities that will exploit a corporation's database. Complicating this situation is many times malicious users know about exploits before the good guys do.

Exploits pose a time sensitive threat to a corporation. The corporation must address and add compensating controls for an exposure until they can remediate the exploit through patching or configuration changes. Deploying a patch into an environment is often slow and involves negotiation with business owners. Downtime due to patching and testing at critical business cycles can result in lost revenue; conversely, leaving an exposure can result in loss of data, reputation, and outages. This is a classic case of risk avoidance and analysis resulting in the security or risk team recommending path of action to the business. Security staff should monitor security bulletins such as the SANS Internet Storm Center (https://isc.sans.edu/) for the latest bulletins, and monitor logs for signs of privilege escalation. If there is security bulletins available for the businesses vertical market (e.g. finance, retail, law enforcement), security staff should subscribe and monitor them. (Ottman, 2010)

2.4. SQL Injection

SQL injection is an older, but one of the most common attack methods used by malicious users to exploit Web applications. In the second quarter of 2012, SQL injection attack rose by 69%. This statistic proves SQL injection is just as prevalent today as it was several years ago. (Ashford, 2012) SQL injection allows the attacker to influence SQL statements that do not correctly filter input from applications to a back-end database. When an attacker is using SQL injection, they try to exploit the syntax and capabilities of SQL to exploit vulnerabilities with the goal of gaining access to a database and data. SQL injection is the result poor input validation of user input and the use of unintentional interpretation of metacharacters by the database. The most common targets for SQL injection are stored procedures and input parameters. SQL injection allows an attacker unauthorized and many times unrestricted access to the database. Using SQL injection an attacker can steal, damage, and modify the contents of database. SQL injection many times is the result of poor design on the part of the development or database administration teams. Applications can over the exposure and risk of SQL injection by implementing input validation and having a mature SDLC (Software Development Life Cycle) process. (Clarke, 2009)

The main consequences of SQL injection are violations of confidentiality, authentication, authorization and integrity. SQL injection can violate confidentiality of data since the data stored in a database is usually sensitive. SQL injection can cause authentication problems if SQL statements vulnerable to SQL injection authenticate users. These statements can allow unauthorized access from one user using credentials without knowing the password for the credentials. SQL injection jeopardizes data integrity. Many SQL attacks allow the unauthorized manipulation of data. (OWASP, 2012)

Most people associate SQL injection with Web applications, but SQL injection can be present in any application architecture. In three-tier architecture, the Web server has a connection pool into the database resulting in logins occurring at the

application layer and not the database layer. Client server architectures login to the application is logging into the database, in this situation many of the SQL injection attacks do not apply. Because Web applications are available to a potentially large number of internal and external users, security programs should consider SQL injection attacks beyond Web forms. The availability of Web applications externally increases the threat and data exposure, taking steps to prevent SQL injection is necessary. (Natan, 2006)

2.5. Weak Audit Trail

Databases contain sensitive information from a corporation, having an audit trail of sensitive, unusual and privileged transactions should be a critical component of any database environment. Not having a reliable audit trail of sensitive, unusual and privileged database transactions is a serious threat to an organization. Each record breached record costs around $200 U.S. dollars for processing. (McKendrick n.d.) In the event of a data breach, being able to identify the records involved in the breach can save a corporation millions of dollars and helps prevent the loss of brand and reputation. Not having an audit trail of data can jeopardize compliance programs that must comply with government regulatory requirements. The audit trail should be robust enough to indentify users, statements and responses. During the SDLC (Software Development Life Cycle), corporations should identify sensitive data, transactions and privileged accounts. Audit trails may be the last line of defense if an attacker can circumvent other security controls. Although audit trails are after an attack and do not prevent attacks, they are critical to any forensic investigation due to a breach. Additionally, audit trails have operational benefits when there are application issues requiring a more intensive debugging effort. Audit trails can help identify difficult problems. (Shulman, 2006)

2.6. Denial of Service (DOS)

Denials of Service (DoS) attacks are old school attacks that never seem to go away and remain a serious threat to organizations. A Denial of Service is an attack on a computer or network with the intention of making that computer of network unavailable to user. DoS attacks can happen at several layers such as network, application, database, and operating system. When dealing in the area of databases, these attacks come in the

form of attackers crashing services, corrupting data, and consuming resources such as disk, CPU or network ports. Many DoS attacks have a link to underlying flaws in the operating system or protocol used by the application. There are many reasons for executing a DoS attack, such as extortion, reputation attacks, and attackers looking for credibility, corporate competition, and people with too much free time. Regardless of the intent of a DoS attack, the threat is real and will not go away in the near future. DoS attacks can also be self-inflicted when configuration errors cause a loss of service. For example, if there is a typo and the number of allowed TCP connections is set too low, connectivity will suffer and users will not be able to access the database. A successful DoS attack can affect revenue and reputation of a company. If the DoS attack occurs at a critical business cycle, it can be devastating to a company. (Shulman, 2006)

2.7. Database Communication Protocol Vulnerabilities

One area of concern is the growing number of security vulnerabilities in the communication protocols for a database. One of the most famous protocol attacks was the SQL Slammer worm. The SQL Slammer attack took advantage of a flaw in the SQL Server protocol, resulting in a denial of service. These attacks are troublesome because attacks to protocols will not show up in native database audit trails because native database audit facilities do not cover protocol operations. Tracking down a protocol attack can be tricky and will often require monitoring capability outside of the database area. No vendor is immune to communication protocol vulnerabilities and patching these vulnerabilities may be difficult because of the testing required for patches and potential business interruption for applying the patch. These attacks are dangerous because they can travel through layers of defense and go undetected. (Shulman, 2006)

2.8. Weak Authentication/Password Attacks

Many corporations implement elaborate and effective security mechanisms to protect the environment surrounding a database, only to have the controls compromised by a weak authentication mechanism to the database. In environments where passwords are the only control in place to protect data; the security of data rests of the strength of the password. Instances where accounts are active with default passwords, or the password

used is weak and easy to guess are easy to avoid. There are a few common ways of obtaining credentials for database access. The first is a brute force attack where an attacker will keep trying common credential combinations hoping to find a match. Brute force techniques commonly use tools to automate this process and may implement high-speed techniques such as rainbow tables. Another method results from a user storing credentials in a place where third parties have access to them. Many users do the unthinkable by posting user/password combinations in workspaces so they have easy access to credentials. This allows an attacker to steal credentials without doing any hard work. Administrators often implement monitoring and utility scripts to help automate routine tasks. These script soften contain hardcoded passwords or configuration files that show up in the process table when the script is running, or the passwords are in a file that lacks access protection. Social engineering is a common technique for password attacks, and is often highly successful. Social engineering takes advantage of the human tendency of being helpful. The classic case of a phone call from the Help Desk asking for credentials so they can troubleshoot a critical business issue. These attacks are classic, but still chillingly effective. Finally, there are cases were password for privileged accounts are the vendor default, or they are blank. Originally, SQL Server shipped with the "sa" password blank. The Internet contains thousands of lists of default username/password combinations. Leaving a privileged account such as "sa" with a weak authentication mechanism allows an attack to easily connect and own the database. (Natan, 2006)

2.9. Backup Data Exposure

Many corporations forget about the criticality of backups until there is a need for a restoration of data. Because of the sensitive nature of database data, corporations should take adequate steps to protect backups. It is a common, but unwise practice to leave backup media unprotected and vulnerable to theft from an attacker. In the event a backup tape is lost or stolen, data not encrypted on that backup media is vulnerable. If corporations have a policy of storing backups off-site for protection, and the corporation uses a shipping service, the corporation has no control of the backup media/data when it is in the carrier's possession. Protecting data at rest applies to backups as well as data on

servers. If a competitor was able to obtain a set of backups, a corporation could lose intellectual property, and sensitive data jeopardizing business operations. There should be a corporate policy requiring the encryption of sensitive data on backup media, and a process to implement the policy. (Shulman, 2006)

3. Policy Development

Policy empowers users to do the right thing protecting companies, employees, customers and information. Development of policy is a critical and often an overlooked activity in security programs. Many industries today are subject to federal, local or industry specific laws and regulations. These laws and regulations will drive business and compliance operations for many organizations. The laws and regulations are critical inputs to the development of corporate policy. Technical people love to deal with technology and many times skip the policy that should be driving the implementation of technology. Policies can exist on several levels in an organization; there are regulations, laws, corporate policies, division policies, local policies, issue-specific policies and procedures. Policies and procedures need to be SMART (Specific, Measurable, Achievable, Reasonable, Time-Based). Policies address who, what and why, procedures address the how, where, and when. (Northcutt, 2010)

There will be technical limitations that could prevent the collection of critical information; policy will help us to provide compensating controls so the collection of information is possible. A good example of a policy compensating for technical deficiencies is one that discourages the use of server-based tools such as SQL Server Management Studio directly from the Window's server host SQL server. Of course, there will be times when direct access is a requirement; these situations should be in the policy. This will help the gathering of SQL data over the network allowing the database-monitoring appliance to report on data. RDP uses encryption, so staff connecting to the server presents a blind spot for the database-monitoring appliance.

Development of a policy will drive the business cases and outline what is the organization needs to protect and set a framework for implementing compensating controls. The success of a database security logging and monitoring program will depend

upon having goals and guidelines in place. By first understanding business goals, laws, regulations, and resources that require protecting an organization can create an effective policy and base business cases on all this information. As mentioned earlier, this preliminary work is critical, but often skipped by many organizations. Preparing the foundation will not guarantee success for an organization, but it will better prepare an organization for the work that will build a successful program. In the appendix, there is a sample database policy to help with enforcement of best practices. The policy in the appendix will serve as an example policy for examples and discussions going forward. (Ottman, 2010)

4. Monitoring Methods

There are three methods available for monitoring database activity: native auditing, installation of a host-based agent, and network-based monitoring. The majority of databases offer native auditing so a database administrator can look at security events of the database. Events of interest include logon/logoff, object access, and queries.

4.1. Native Auditing

The benefit of native auditing is this option does not require an additional spend for third party appliances or software for implementing. However, the implementation of native auditing incurs a performance penalty on the database, and the logs produced by native auditing need protection against modification and general user access. The performance penalty incurred from implementing native auditing will be hard to get support from the database administration and application support groups. The implementation of native logging does not require any additional spends, but the logs will require secure, centralized storage and a method of correlating and reporting on events. The implementation of native logging will increase your storage cost, and the cost may be substantial because native logging can produce a high volume of data requiring a large amount of storage to hold these logs. If your organization has data retention requirements, these logs will increase your backup storage cost because of their large size. The audit logs without any reporting and monitoring capability will not add much value to your database security logging and monitoring program.

There are additional issues with native logs that make the performance hit versus benefit not worth the implementation. Native auditing is difficult to deploy and manage on a large scale; in the end, the logs do not provide as much useful detail as network-based logs. This makes correlation of database activity with other infrastructure activity difficult and very laborious. If there is a network-based attack in the logs, it is difficult to monitor the response and content of the queries in native log files. During a breach investigation, knowing the response to a query can help determine the extent of the breach, and the financial obligation of the corporation for the breach. (Ottman, 2010) If a company can prove the exposure of only X number of records during a breach, this gives the corporation a better idea of the extent of financial liability for the attack. Every record breached will cost around $200 U.S. dollars to contact the owner of the record. (McKendrick, n.d.)

If users access the database via a service account, the logs will not specify the actual user who initiated the database action; the logs only show an action taken by a service account. Tracing pooled connections back to the original source user is not possible only using native database logs. If the database contains sensitive data such as social security numbers, there is a possibility that information may end up in the native log files. This complicates compliance issues because sensitive data is now stored in an additional place, and one that lacks security controls. Native logs store all log data together increasing the difficulty of separating users; if there is a need to separate users, it is difficult because the logs store all users together. The performance penalty coupled with the other issues discussed does not make a good case for using native logging in a database environment. (Ottman, 2010)

4.2. Host Agents

Vendors selling database monitoring products typically offer two solutions: agents (host-based) and network-based monitoring. Database agents reside on the database server and monitor memory or transaction tables for SQL transactions. Some database agents require enabling of some native auditing features so they can retrieve audit data. As discussed earlier, enabling some or all of the native database auditing adds a performance penalty to the database.

Agents are programs installed on the database server running in a privileged mode as a service or daemon depending on the database platform. These programs need to run in a privileged manner and normally have hooks into the kernel so it can intercept SQL traffic and send the data back to a monitoring appliance. Because these database agents run on the database server, they are able to capture all database activity, this includes local activity from users directly connected to the server. Most agents require applying a policy to the agent so it knows what to monitor. This allows for custom policies based on the business needs of the database.

Network-based only solutions can only report on activity coming from the network. With network-based monitoring, if the communication to the database is via an encrypted communication and the database-monitoring appliance does not have the private key, it cannot decrypt the traffic to monitor it. Host-based agents are able to monitor all database activity since they are resident on the database server. Although this sounds like an ideal solution, it does have several drawbacks. Because the agents need to run with privileges and have access to kernel resources, they introduce additional risk to the server in terms of potential vulnerabilities. If the privileged program has logic or coding errors, it is possible the database server can suffer a DOS (Denial of Service) attack from a program that is helping to protect the resource. This is a bit ironic. Host-based agents allow the ability to drop connections that appear to be malicious in nature, functioning like a host-based IPS (Intrusion Prevention System) for databases. Although having the ability to stop malicious traffic may seem attractive, this has the ability to prevent legitimate connections and have the database acting like a BPS (business prevention system), performing a DOS attack on clients.

The audit logs produced with the host-based agent are inferior to network-based monitoring because they lack the response of the queries. The data has little organization because all the records are together and it can be laborious. The biggest issue with database agents is the on-going maintenance. Because the agents have ties to the kernel and database revision, changes to the database version or operating system may require an update to the agent. If the database environment is large, this is very time consuming and may require a visit every server to update the agent. Some solutions do have the

ability to update agents from a management station, but this requires elevated privileges to the server from another resource. This does pose additional risk by adding additional access vectors to the database server. The database team will likely disagree with an outside party have elevated rights into the database. Regarding responsibility for database agent maintenance, there may be three teams involved, each with different interests and responsibilities. The server teams are responsible for the maintenance, patching and administration of the base operating system. The database team is responsible for the maintenance, patching and administration of database software. The security team is responsible for making sure the agent version matches the operating system release and mode (64-bit versus 32-bit), as well as the database version. In separation of duties, the security team would have one of the other teams that would update the agent as needed. The agent will place a greater need of communication and cooperation between the teams, if one of the items does not match, the agent may not function, or worse cause stability issues for the service provided by the database. Agents will also place additional loads on the server, although the load will be much less than the load resulting from native logging. If there are issues with the server or database, the database agents will always be one of the first suspects during troubleshooting. If your business needs require an agent, it is critical they agent undergo strenuous testing giving infrastructure teams and management confidence in the technology. (Ottman, 2010)

4.3. Network Monitoring

Database monitoring from the network, referred to as DAM (Database Activity Monitoring) has been around for about 10 years, and it allows corporations to perform better analysis of database traffic without causing a performance problem on the database server. The view of SQL traffic on the network allows for better behavior monitoring without increasing a load on the server. Network monitoring requires an appliance or the use of span ports on a switch to capture traffic. Span ports on a switch mirror all the traffic on a switch and make it available on a port for use. Appliances such as a Gigamon are high-density intelligent traffic visibility nodes that replicate and filter network traffic to monitoring and security tools. This allows the analysis of traffic without affecting network performance and the user experience. There are many advantages with using a

network-based monitoring solution for database activity monitoring. For example, monitoring database traffic over the network makes it is possible to find new database instances in the environment. Monitoring for new or rogue database instances allows for a tight integration with corporate change management and security policies. This advantage of detecting rogue database instances is difficult to accomplish without using network monitoring. It is very likely host-based monitoring would not detect any new database instances.

Analyzing network communications can reveal issues with simple password use and consumers of sensitive data. Knowing who is using your data and where it is going helps with policy enforcement and breach disclosure. Network monitoring gives the unique view of stimulus and response of queries, detecting large amounts of data leaving the server from a query. If corporate policy does not allow the transfer or use of production data in non-production system, network monitoring can help detect staff moving data from production to non-production. An attacker probing for SQL injection vulnerabilities can send statements in a very low volume manner trying to avoid detection. Native auditing and host-based agents have difficulty detecting this activity because it gets lost in the logs and does not report on the stimulus response aspect of the query. The network-based monitoring is ideal for detecting these types of stimulus/response probes.

Most network monitoring functions can act in a protection mode, working like an IPS (Intrusion Protection System) for databases, although this is dangerous. Based on certain criteria, the database monitor can send a TCP reset to clients when it detects suspected malicious behavior. This allows the database monitor to run in an IPS-like mode, however if there is something abnormal about the design of an application in your environment; this could result in a DOS (denial of service) attack against users of your database application. This may result in a career-limiting move. There is one issue that can be frustrating for database security logging and monitoring, that is the use of secure protocols. Staff connecting to a server via SSH or RDP using encryption, the protocol encrypts the traffic and renders the network monitoring useless because it cannot see inside the tunnel without having a copy of the private key.

Security professionals should commend staff for using encrypted protocols; however, network-based monitoring is not effective when dealing with encryption. There are business needs requiring staff need to perform actions directly from the server; this is where policy and additional controls can help. If the encrypted protocol uses a certificate, it is feasible to share the private key with the database-monitoring appliance and decrypt the traffic sent over the network. (Natan, 2005)

4.4. Monitoring Conclusion

There are three methods of monitoring discussed here: native, host-based and network. Although native auditing is available without any additional spend for monitoring tools, the performance penalty, resource requirements (CPU/disk), and the information available in the logs do justify using this method for a security program. Host-based agents are better, but still there is a penalty with CPU utilization and the total cost of ownership makes this option less desirable. There are cases where data is so critical that using a host-based agent along with network monitoring is a requirement. Sensitive financial transactions such as stock trades and monetary transfers between organizations are examples. The most organizations using network-based monitoring is the best fit from a cost, control and compliance standpoint. For purposes of this discussion, the fictional organization will chose to implement a network-based monitoring program.

5. SQL Server Recommendations

Microsoft SQL Server is one of the most popular database platforms deployed for database management in business. Microsoft SQL Server is easy to implement, powerful, and a good fit for many organizations that implement Microsoft tools. Beyond SQL Servers business use, many appliances run on the Windows Operating System using Microsoft SQL Server to store data. Understanding SQL Server is a marketable skill and a technology many security professionals will encounter routinely. The following recommendations are best practices that will help provide compensating controls for SQL Server environments.

5.1. Monitoring of Windows Event Logs

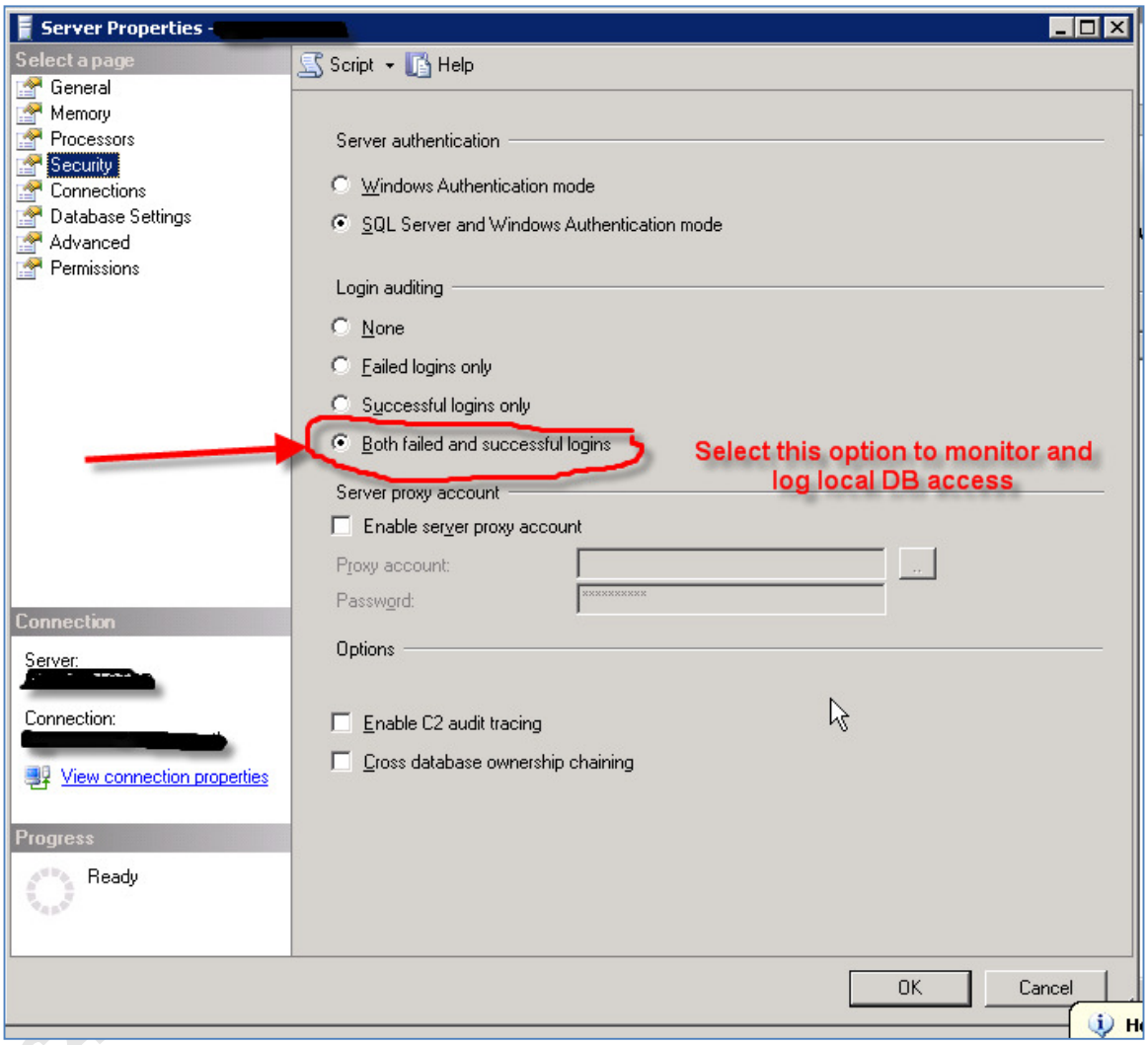
A regular review of Windows Event Logs should complement the monitoring and logging of SQL Server databases. The Windows Event Logs of the Operating System hosting the SQL Server database need regular review and monitoring. In the common vulnerabilities section, there were several common vulnerabilities where regular Operating System log review will help detect issues. Security related events commonly detected in Windows Event Logs are Database Platform Vulnerabilities, Weak Audit Trails, Denial of Service, Database Communication Protocol Vulnerabilities, and Weak Authentication. Traces of attacks in these areas may show up in the Window's Event Logs. For the most critical database systems, consider monitoring critical system directories, database configuration files, backup directories and the system registry. The operating system's audit policy should match corporate policy enabling the proper amount of auditing on the server. One setting to be careful about enabling is "Audit: Shut down system immediately if unable to log security audits policy" which allows users to start a denial-of-service attack through the audit policy. Carefully research, understand, and test any audit setting modified on the server before implementation in a production environment. (Microsoft, 2005)

A great resource for researching Window's Event codes and their meaning is Randy Franklin Smith's Ultimate Windows Security Web site at http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx?i=j. This Web site contains a comprehensive list of Event codes for various Windows' operating systems along with when they appear in the Event Log. This is necessary have resource for deciding which Window's Event code are important to a monitoring program.

5.2. Configure SQL Server to Log Success and Failed Logins

The default security setting on SQL Server is to log failed logins to the Window's Event Log. Modifying the SQL Server setting to log both (local) failed and successful database logins to the Window's Event Log will help increase the visibility into who is using the database. Using the recommendation from the previous section of reviewing

Window's Event Logs on a regular basis, this will help with auditing and monitoring of local access to the database. Not changing the default setting of logging only failed logins will result in an audit gap when users connect directly to the server using tools such as Dameware and RDP, and then use a tool local to the server for database access. This database access locally from the server will go undetected. Tools like Dameware and RDP use encryption when establishing connections between end-user computers and the database server, rending network-sniffing technologies useless. This simple change allows for the reporting of local database authentication events when users connect to the database locally from the server using an encrypted channel. It is important to note if there is not an agent or additional logging on the database, this setting will detect local access, but the actions while connected locally will go undetected without additional compensating controls. Below is the SQL Server configuration screen and Windows Event logs for successful and failed local logins. (Chapman, 2011)



Configuration screen for SQL Server to enable success and unsuccessful logins

Successful Login Event:

```
Event Source:  MSSQLSERVER
Event Code:    18453
Type:          Audit Success (4)
User Name:     corporation\gooduser
String[%1]:    corporation\gooduser
String[%2]:     [CLIENT: 10.10.10.10]

Login Success:
Computer Name: SQLPROD
Event Source:  MSSQLSERVER
Event Code:    18454
Type:          Audit Success (4)
User Name:     N/A
String[%1]:    companylogs
String[%2]:     [CLIENT: 10.10.10.20]
```

SQL Server successful login – Windows Event Log

Failed Login Event:

```
      Computer Name: sqlserver1p.goodcompany.com
      Event Source:  MSSQLSERVER
      Event Code:    18456
      Type:          Audit Success (4)
      User Name:     corporation\sqlagent
      String[%1]:    corporation\sqlagent
      String[%2]:     Reason: Failed to open the explicitly specified
database.
      String[%3]:     [CLIENT: <named pipe>]
```

SQL Server unsuccessful login – Windows Event Log

5.3. Monitor Active Directory Authentication and Service Requests

A security best practice is monitoring of Window's Event logs for security and operational events. Many corporations overlook the value of monitoring Active Directory Kerberos ticket requests and authentications. These events can be very helpful in tracking SQL activity and authentication to the SQL database. In an environment where an SQL Server monitoring is via network sniffing and the traffic is encrypted, monitoring Kerberos requests can assist in auditing which credentials and processes are attempting connections SQL Server. If there are problems with encrypted connections because of using Kerberos authentication, the Kerberos granted tickets could help with credentials accessing a SQL Server database using Windows Authentication.

Authentication Ticket Granted

Authentication tickets are useful for tracking initial logins. These Windows event are the result of a user contacting the domain controller and requesting a Ticket Granting Ticket (TGT). If the credentials are correct on the domain, and there are no restrictions on the account, the domain controller grants a TGT and logs a 672 event in its event log. The User field always reports the user as *SYSTEM*, so administration staff will need to correlate IP addresses to user. If there is a '$' after the computer account's name this indicates the event is computer generated. These events can help trace users when monitoring systems cannot decrypt login packets because of Kerberos authentication.

```
Jul 16 06:59:59 domaindc1p EvntSLog: Security,672,Mon Jul 16 13:59:59
2012,NT AUTHORITY\SYSTEM,9,MY_SERVER;FOOBAR;%{S-1-5-21-2130074997-
794050694-931750244-17949};krbtgt;%{S-1-5-21-2130074997-794050694-
931750244-502};0x40810010;-;0x17;2;10.10.10.10;;;;;;
```

Service Ticket Request

Service ticket events are the result of a computer or a user accessing a computer on the network. While event Authentication Ticket Granted (event id 672) recorded initial logins, the event Service Ticket Request (event id 673) is the result of a computer or user accessing a server on the network. This is an event only available on domain controllers, but is very helpful in determining whom and what is accessing a server. If an attacker is trying an attack by mounting a drive, the request to the operating system is available with this ticket.

```
Jul 16 09:59:58 domaindc4p EvntSLog: Security,673,Mon Jul 16 13:59:58
2012,NT AUTHORITY\SYSTEM,9 ,jim@foobar.com; FOOBAR.COM;servername$; %{S-
1-5-21-2130074997-794050694-931750244-62310};0x40810000;
0x17;10.10.10.20;-;{eadc3444-d170-97d3-47ac-50b5350699a6};-;;;;;;
```

Ticket Granted Renewal

This ticket may be useful in only a small number of situations and environments. Kerberos has limit on a ticket; this requires a renewal of the ticket after expiration. After Windows 2000, the server will automatically contact the domain controller and renew the ticket. This creates a 674 event in the domain controllers Event log.

```
Jul 16 09:59:44 domaindc2p EvntSLog: Security,674,Mon Jul 16 13:59:44
2012,NT AUTHORITY\SYSTEM,9 ,jim@foobar.com; foobar.com;krbtgt;%{S-1-5-
21-755119451-1620593998-612134452-502};0x2;0x17;10.10.10.10;;;;;;;;;
```

Pre-authentication Failed

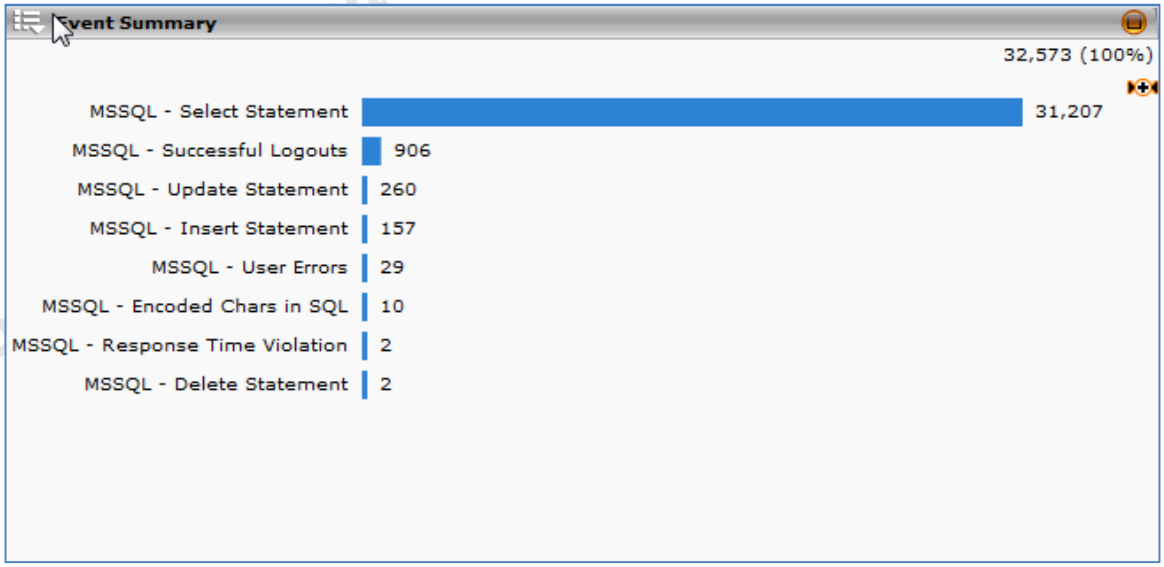
A pre-authentication failed message is interesting in every environment. If a user tries to connect with a valid username and incorrect password, the domain controller will log a 675 event in the Event Log. This record contains the username, domain and IP address where the login failed. This Window's event can track malicious users trying to connect to a SQL Server by brute force once they have a valid username.

```
Jul 16 09:59:54 domaindc3p EvntSLog: Security,675,Mon Jul 16 13:59:54
2012,NT AUTHORITY\SYSTEM,9,jim;%{S-1-5-21-2130074997-794050694-
931750244-90060};krbtgt/foobar;0x0;0x19;10.10.10.10;;;;;;;;;;
```

(Microsoft, 2000)

5.4. Deploy Certificates on SQL Servers

SQL Servers using Windows authentication will have encrypted login packets due to the use of Kerberos. If monitoring of the database is via the network, the monitoring system will miss most of the logins occurring at the server. Typically, the monitoring system will report a large number of SQL Server logouts and only a few SQL Server logins. Deploying either self-signed or PKI certificates and sharing it with the database-monitoring appliance can be helpful in helping track user authentications and decrypt any SQL traffic that may leverage encryption. SQL Server generates a certificate when it starts and shares the public key with any domain controllers. Applications using integrated Window's authentication send login packets encrypted due to the Kerberos requests. The encryption of login packets is good for security, but creates an audit gap in a Security Logging and Monitoring program because of the inability to track user credentials when applications authenticate. Here is an example from a database-monitoring appliance showing SQL activity for a day.



SQL Server daily activity high-level report

This example shows 906 logouts without any login activity. The encryption of login packets creates an audit gap where network-based monitoring misses the login packets. (Microsoft, 2010)

5.5. Use Policy to Limit DBA Privileges

Create a database policy based on business requirements and one that aligns with corporate policy that will grant the minimal amount of privilege users and database administrators need to perform their job functions. This is the security principle of least privilege. The database administration staff should never have Window's System Administrator privileges for the host. Consider having two accounts for each database administrator, one privileged account and one for performing daily duties. As a step to further control access to sensitive information, privileged accounts can act like accounts governed by a fire call process, where a ticket or request is a requirement for each use of the account. This will not win any friends with the database administration staff, but it will help to report on privileged versus non-privileged use by staff, and discipline staff to use privileged accounts when necessary. The database policy can help enforce disciplined data access and use of privileged accounts.

5.6. Use Policy to Limit RDP Access to the Host

RDP can create audit gaps if not addressed by policy. It is advisable to create a policy limiting direct access to SQL Server host with tools like Dameware or RDP, and routinely monitor RDP access using the Window's Event Logs. If a database administrator needs direct access to the Window's server, they should have minimal rights that match their job responsibilities. They should never have system administrator privileges. RDP ability to a server should be limited to staff who require it for their job responsibilities. RDP creates an encrypted tunnel for the user allowing them secure access to the server. Although this is a great security practice, it will cause gaps when users connect directly the host operating system and access SQL Server. If the database contains highly sensitive information, it may necessitate the deployment of a database agent. The additional maintenance and cost of agents many times make them unfeasible. Monitoring of RDP access via the Event Log coupled with *firecall* process for requesting and granting RDP access can help deter unnecessary server access and create greater visibility into who is using RDP.

5.7. Create Policy to Help Govern Database Access

Create a policy that will encourage database administration and access over the network, while limiting direct server access to an only-when-necessary situation. Having a policy in place recommending database administration staff use network-based tools will help increase visibility of database activity because local access of the SQL Server will only happen when necessary. Cases requiring direct server access would be patching and routine maintenance. The situations requiring direct database access would be associated with a change ticket creating an audit trail for the activity. Forcing staff to use network-based tools may remove the need and added cost and maintenance of database agents. This will help ensure the monitoring of SQL activity by an appliance using network traces, without having to rely on a host agent to monitor access. RDP uses an encrypted channel so a database appliance cannot monitor the SQL activity. In addition to creating a policy, monitoring Window's Event logs for RDP access should also be a routine activity. Routine monitoring of server access via RDP will help report on policy violations, and show approved RDP access for non-routine activities.

5.8. Avoid Deploying Database Agents

As discussed earlier, using agents on databases hosts increases the cost of a Security Logging and Monitoring program. The agents need to match the version of the operating system (32/64 bit), database versions, and possibly patch levels. Additionally, agents increase the resource demands on the database server and require additional patch cycles when the vendor provides a patch or rule updates. Adding agents to a server increases the resource consumption, cost of the server, and the administration overhead for the server. Many organizations need several additional agents for business functions such as operational monitoring, backups, performance, etc. Administration teams start joking servers run more agents than business software. There are cases of critical database servers that present a business case that warrant the deployment of a database agent. For these cases, organizations should perform a cost, risk, benefit analysis. (Ottman, 2010)

5.9. Database Hardening and Stored Procedures

Hardening the database combined with intelligent use of stored procedures can protect and limit damage from SQL injection attacks. Applications using the database should use low-privilege accounts to authenticate into the database, and should observe the principle of least privilege. There should be a removal or remediation of any unnecessary access to the database. This should include the removal of default accounts that are part of the product base, but not required for business operations. As part of the hardening process, database administrators should remove stored procedures that are not required for the business function of the database. Removing unnecessary stored procedures can remove some potentially vulnerable code and additional exposure. There is best practice hardening guidelines available on the Internet such as The Center for Internet Security. (Chapman, 2005)

Stored procedures can help mitigate and lessen the severity of SQL injection attacks. Stored procedures are subprograms available to applications that are stored in a database. A security benefit of stored procedures is the ability to grant an application access to execute a stored procedure, but not allowing access to database tables. This increases the security of the database because if an attacker can perform SQL injection through a stored procedure, they do not have access to tables and should reduce the exposure of the attack. When developers implement stored procedures, they should avoid using dynamic SQL within stored procedures. Using dynamic SQL allows the concatenation of strings exposing the stored procedure to SQL injection. Every parameter passed to a stored procedure should function as a parameter in the stored procedure. When implemented properly, stored procedures treat parameters correctly handling special characters treating them as text. (Clarke, 2009)

6. Database Encryption

The topic of database encryption is often misunderstood and the implications of implementing database encryption must take into consideration the total cost of ownership. One side views database encryption as the best way to lock down and protect sensitive data, while the other side sees database encryption as a production burden.

Arguments for and against database encryption have valid points. Encryption increases costs through increased consumption of server resources such as CPU and disk and proper management of keys is crucial to the success of the program. Most major database vendors provide encryption as a built-in feature because compliance and regulations require it. On the surface, encrypting a database may seem like an easy decision, but reviewing the cost of implementing database encryption often does not justify it. The performance penalty of using database encryption coupled with the management of encryption programs is complex and can quickly become a support nightmare especially if encryption spreads throughout the enterprise. Many times this combination will override and justification of using database encryption. (Ottman, 2010)

6.1. Storage versus Transport Encryption

Users, developers and managers without an understanding of what they are talking about bandy about encryption. It is important that there is an understanding between the distinctions of storage encryption versus transport encryption. Encryption can ensure the confidentiality and integrity of data while in transit and while at rest, depending on the implementation. SSL will provide confidentiality of data while in transit; SSL does not protect data while at rest. Many people have a misconception about SSL believing the use of SSL protects data in transit and while at rest. This is wrong. Encrypting data in transit using SSL is more on the application end; database encryption protects data at rest before it starts the journey to the end-user. The encryption of data on disk provides protection against unauthorized access and data integrity. The goal of database encryption is protection of data while at rest. Database encryption does not protect data from compromise or abuse within applications. Without the use of encryption, data at rest is vulnerable to attack, exploits and privilege abuse. Many exploits attack services that run with privileges and once successful have access to sensitive information stored at rest on the server. Alternatively, exploits may attack a non-privileged account and then elevate its privilege once successful resulting in access to sensitive information. Finally, there is the threat from staff who abuse there privileges and access confidential data for non-support issues. Encrypting data while it is in transit,

and while at rest will protect it, but it does not guarantee the data is safe from malicious users because of the amount of attack vectors. (Natan, 2005)

6.2. Encryption Planning Areas

An organization must carefully examine the business drivers and implications of implementing database encryption. Organizations must analyze four areas before implementing database encryption. The first and most important issue of database encryption deals with management of keys. This is critical to the success of a database encryption program; the encrypted data is useless unless there is a way to decrypt it. Organizations must have policies and procedures that ensure normal business operations, backup, and recovery with effective key management. Mistakes made with management and operation of keys pose business continuity and data loss risks for organizations. Organizations should regularly test key management programs in a variety of situations to verify can operate under a variety of conditions. If an organization loses its encryptions keys, it will not be able to decrypt its data leaving the data useless and exposing the organization to a shutdown of business. Organizations with multi-vendor platforms have and even more difficult task since each database vendor has a vendor-specific implementation of native encryption. The second area of concern with database encryption deals with application transparency. An obvious point is applications need to read and write data or the application will not function. Encrypting columns in the database will likely incur a performance penalty in applications. Third, organizations must have policies and procedures in place that drive the use and lifecycle of keys and encryption. Adopting an encryption policy will affect the handling of data throughout its lifecycle and use. The policy will help direct data usage, backups, recovery, business continuity planning, and disaster recovery. All business lifecycles must align with the policies and procedures. Finally, there is the area of performance. Business is very competitive, and in the current society of "got to have it now," performance and poor end-user experiences can kill a business. Encrypting primary keys or index fields will result in performance problems because it will affect sorting and the speed of queries. Performance issues that affect the user experience are tough to sell to management and application owners. If a performance issue is serious enough, it can affect the revenue

and reputation of a corporation. Who wants to do business with a WEB site that is slow and time consuming? Encrypted database columns increase the amount of space required for storage and the load on the CPU. Organization must investigate which data (columns/indexes/tables/etc.) are good candidates for encryption and understand the total cost of encrypting that data. Rushing the implementation of using database encryption without researching, testing, and preparing is a recipe for failure. (Ottman, 2010)

6.3. Key Types

Keys are mission critical for database encryption. SQL Server has the ability to use three different types of keys, symmetric, asymmetric and certificates. The type of key used depends on the business needs, application requirements, tolerance of disk bloat and performance needs. (Microsoft, 2012)

6.3.1. Symmetric Keys

Symmetric key encryption uses the same key to encrypt and decrypt the data. Symmetric keys can be difficult to secure because the shared secret encrypts and decrypts the data, securing the key with SQL Server with appropriate controls in place is the recommendation from Microsoft. Symmetric keys offer better performance than certificates or asymmetric keys. Symmetric keys are a good solution if encryption of data while it is in the database is the only encryption requirement. If data retrieved from the database can be stored and retrieved as clear text, symmetric keys are a good solution because encryption and decryption using a symmetric key is fast.

6.3.2. Asymmetric Keys

Asymmetric has two different keys, one to decrypt the data, and one to encrypt the data. Asymmetric encryption consumes more resources than symmetric encryption, but they provide better security than symmetric encryption. Asymmetric keys do not use a shared secret, but they do require the owner of the private key keep the private key a secret. Often symmetric keys use an asymmetric key for protection prior to storing the symmetric key in a database. Asymmetric keys are a good solution when there is data sharing between applications and the SQL Server database. Applications use the public

key to encrypt data sent to the database, SQL Server protects the private key used to decrypt the data.

6.3.3. Certificates

Certificates are another form of asymmetric encryption that is a digitally signed statement that binding public and private keys to an identity. Certificate work on trust relationships, if Jim trusts Rodney and Rodney trusts Joe, and then Jim will trust Joe. Certificates have an expiration date allowing organizations to expire certificates breaking the bind between a public key and the identity. Certificates are a good choice when there is a need to use asymmetric encryption and a need to associate the key with a user or group. For example, if there is a need to encrypt and decrypt data externally to an organization, a certificate will help identify who encrypted the data. (Keily, 2006)

6.4. SQL Server Encryption Levels

SQL Server implements two levels of encryption database-level and cell-level. Microsoft refers to entire database encryption as Transparent Data Encryption (TDE) and cell-level encryption when encrypting individual columns. Cell-level encryption allows for greater control over the data targeted for encryption, compared to entire database encryption TDE. Data encrypted at the cell-level retains its encrypted state in memory unless it is actively decrypted. The encryption algorithm uses a salted value, meaning the encrypting the same data twice will produce different results. This prevents analysis of the ciphertexts, but prevents efficient searching of the encrypted data. Encryption of data introduces randomization, making it impossible to index data, causing performance problems because of the issues associated with using encrypted values for searching. There are performance issues when a database encrypts the primary key or index field in the database. Cell-level decryption requires data analysis on the data targeted for encryption. Finally, cell-level encryption functions have a limit of 8000 bytes for the data they return. (DamirB, 2012)

TDE provides a method of encrypting data without having to make application changes, and provides a way to encrypt the entire database. The database and log files are encrypted while at rest on the disk, and are decrypted when read into memory. TDE encryption happens at the page level before writing data to disk and decrypts data from

disk when it reads it into memory. TDE does not add additional bloat onto the size of the already encrypted database. TDE uses an asymmetric key stored in the database boot record called a database encryption key (DEK). SQL Server secures the DEK by using a certificate stored in the master database protect by an Extensible Key Management (EKM) module. TDE can help protect data during its lifecycle in periods when the data is vulnerable such as on backup tapes. (Cristofor, 2007)

Encryption is CPU intensive and requires careful planning through a business review prior to implementation. The business view will need to weigh factors such as regulations and local policies along with the expected affect encryption will have on the end-user experience. Developers should avoid encrypting primary keys and index fields because it seriously affects the sorting of data and will increase the response time from queries. Encrypting the database can increase its size from 3% to 5% of its original size. (DamirB, 2012)

6.5. Key Management

Database encryption can add an additional layer of defense for sensitive data, but before the implementation of encryption, organization must understand the effect encryption will have on daily operations. The most important topic concerns the location and storage of keys used for the encrypting and decrypting of data. As part of the key management program, there should be logging and review of key access, as well as a logging and review of the protection mechanism of the keys. Key management is complex and requires a thorough understanding of the life cycle, protection, and management of keys. A complete program has a plan for recovering lost keys and a plan to recover data if the keys are lost or retired. An overlooked area is the recovery of old data where the keys may no longer be in production. Being able to recover old data should be part of a business continuity plan. If an organization has a Public Key Infrastructure (PKI), is it possible to manage the database keys with the existing PKI? A PKI system can help protect and manage the database keys. One important item to thoroughly research is how encryption affects backup and restores. If the data is important enough to warrant encryption, the backups should also leverage encryption. This will protect backup data shipped off site. If there is a policy to change keys

periodically, there needs to be provisions made to safeguard old keys allowing the restoration of older backups if needed. There are cases of organization exposure because keys were lost for backups, or there was a need for restoring a database and it failed because the key were lost.

There are many issues surrounding encryption that must align with policies and procedures before an encryption program starts. Database encryption is not trivial and requires careful thought and planning before implementing. Corporations should have policies and procedures that align with key management, and there should be a thorough data study to determine if encryption is beneficial and worth the cost. If there is a need for further defense-in-depth measures, encryption can help with careful planning. Encryption is CPU and disk intensive activity that requires thoughtful design prior to implementation. Having a process to track access to keys and a regular log review of the access should be part of the data encryption program. Verify there is a real business need for using data encryption and limit the amount of data encrypted.

7. Use Cases

This section will address issues from the top vulnerabilities business cases discussed earlier. The business cases will focus on creating controls to address the top database vulnerability list presented earlier. The controls used, will be a combination of policy and technical controls. Applying the proposed controls will not remediate every vulnerability or exposure from happening, but it will increase the security posture and give better visibility into the environment. The compensating controls should follow the SMART criteria: specific, measurable, attainable, relevant, and time-bound. The compensating control used for addressing each issue will balance cost, risk and reward. Not every solution will result in a technical control stemming from a database-monitoring appliance – each solution will balance cost, risk and reward.

7.1. Privilege Use and Abuse

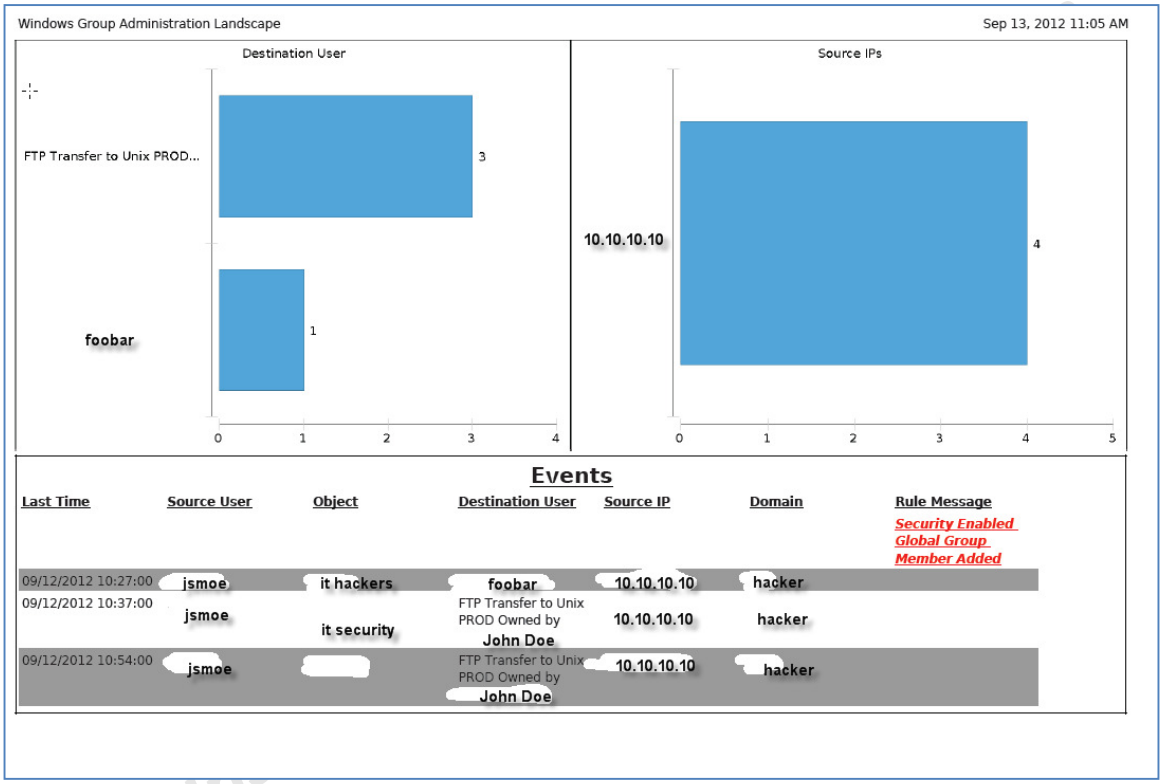
The first three business cases deal with use and abuse of privileges. This section will address excessive privilege abuse, legitimate privilege abuse and privilege escalation. Excessive privilege abuse is often the results of user accounts having

privileges beyond their job responsibilities. Many times the staff responsible for assigning user privileges will grant generic access for all users because the database staff lacks the time to define and update granular privilege control. Excess privilege and lack of privilege control is often the result of ignorance or laziness concerning the understanding of security models, job responsibilities and data access. Security models, job responsibilities and data access is a complex subject and typically spans departments and disciplines. The general category of privilege auditing will benefit from additional processes to augment a security appliance. Authentication and group association modeling is complex, and the relationship between the operating system and database security models adds complexity to already difficult subject. It is important for corporations to have an understanding of security models and data relationships along with a documented procedure for approving the granting and revoking of privileges. (Natan, 2005)

The best solution for dealing with the possible damage from excess privilege is to restrict access with query-level access control by restricting SQL and data operations to the minimum-required. This is a case of implementing the principle of least privilege. The granularity of data control must extend beyond tables to specific rows and columns in a table. This will help detect accounts that try to access data beyond their privilege level. Implementing this principle will help solve many of the common database vulnerability problems. Deciding what users and groups get what access should take place during the design and architecture phase of the SDLC. This will provide developers and administrators a blueprint for the access levels to the data. Once the application is in production, keeping control of data access is much easier and consistent because everyone knows what to do. The principle of query –level access will appear in several of the other vulnerability discussions. (OWASP, 2012)

Creating entitlement reports is another control used for auditing access to the applications and databases. If the entitlement needs management approval through a ticketing system, adding, suspending, deleting or modifying access should show up in the entitlement report. A compliance department can validate the requests and approval for each entitlement added, revoked or modified. When there are discrepancies between the

entitlement report and requests, the discrepancy will require investigation and correction. Following is an entitlement report.



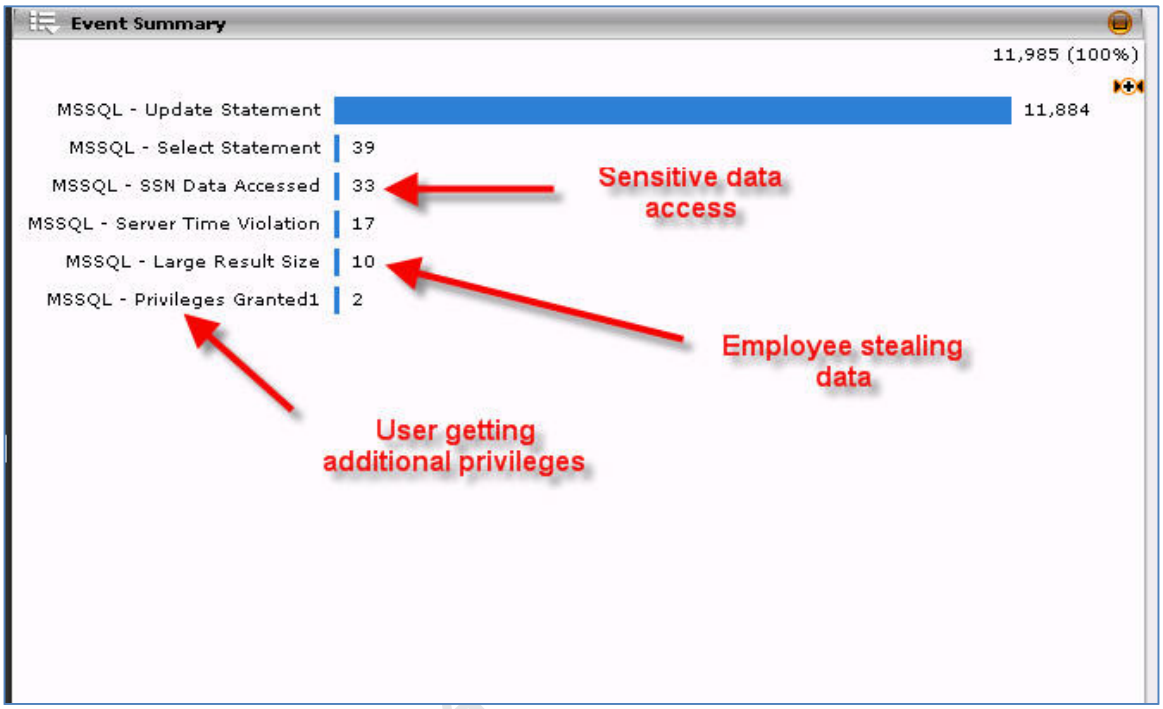
Entitlement Report

Monitoring for legitimate privilege abuse can be more difficult than excess privilege. Some jobs require a high degree of privilege to get the job done; database or system administration is a great example. System administrators need universal system access, and by the nature of database work, the database administrative group receives a great deal of privilege. Somebody has to have absolute authority in order to administer the database and operating system. Beyond these infrastructure groups there are employees who require a higher level of privilege to access customer or employee records for their routine duties. Without this access, they cannot perform their job function.

An application may restrict the viewing of customer records to conform to laws, regulations or policy. An employee may realize they can connect to the database with a Microsoft Office tools and download the data to their personal laptop. The employee may view this as a productivity enhancement so they can do more work faster, and in

their mind more efficiently. The employee does not realize the data is vulnerable at this point since there are less controls around access. If the laptop becomes the victim of malware or theft, the data is at risk. Next, there is the employee who is pilfering data for personal gain. There is always the risk of the rogue administrator who grants privileges without justification, or who becomes a data thief. (Shulman, 2006)

Giving business owners, compliance and audit staff the ability to view data access and transactions can help protect data and organizations. The combination of people, processes and technology will become a winning combination in the fight for data protection and legitimate privilege abuse. There is a report below that will demonstrate implementation of a process based on a policy of auditing sensitive data transactions. There are three items of interest in this report. The first is who accessed SSN data in the application. The report indicates the application accessed social security numbers 33 times. This is not 33 records, but 33 distinct transactions. The business owner should know if this within a normal activity and investigate the circumstances if it is above a normal baseline. The second item of interest is a large data transfer that occurred 10 times during this period. Transferring large amounts of data may be normal based on the nature of the application. However, if the business owner sees a number above normal activity this is a reason for concern. The last item on this report is the granting of privileges. This is an administrative action taken by an employee with privilege. The granting of privileges should follow a well-defined process that has an audit trail of approvals with it. The compliance and audit should be able to follow this trail of activity from the initial request to the granting of privileges.



Sensitive data access report

The combination of people, processes and technology presented here will help data and organizations, but it will not provide 100% protection. The malicious employee who flies low and slow under the radar will be hard to detect with almost any control. The process described here will increase the security and audit posture of an organization that needs to address legitimate privilege access.

The third form of privilege abuse is privilege escalation. This is where an attacker or malicious user takes advantage of a vulnerability to elevate privileges from a normal user to that of administrator or privileged user. Database vulnerabilities can be present in many areas such as stored procedures, protocols, SQL statements, vendor code, and operating system functions. These attack vendors are difficult to monitor because they serve the user community with normal operations. Attackers hide in this normal traffic and exploit a vulnerability elevating their privileges. Once an attacker has elevated privileges, they can manipulate and steal data, modify and disable security settings, create or delete accounts, add malicious code, etc. (Natan, 2005)

Using query-level access against the data will help minimize the damage done with some attacks. If an attacker gains root level authority, they have control over the system and query-level access will not help. The operating system and database should

undergo a hardening process from an organization such as CIS, NIST, or OWASP. The hardening process will need to undergo testing to make sure it does not interfere with any business processes or functionality of the application. If the organization has security devices such as an IDS or IPS, these can help detect and possible prevent privilege escalation attacks before they happen. Monitoring access to the database, in combination with monitoring of the operating system logs can help detect some privilege escalation attacks. (OWASP, 2012)

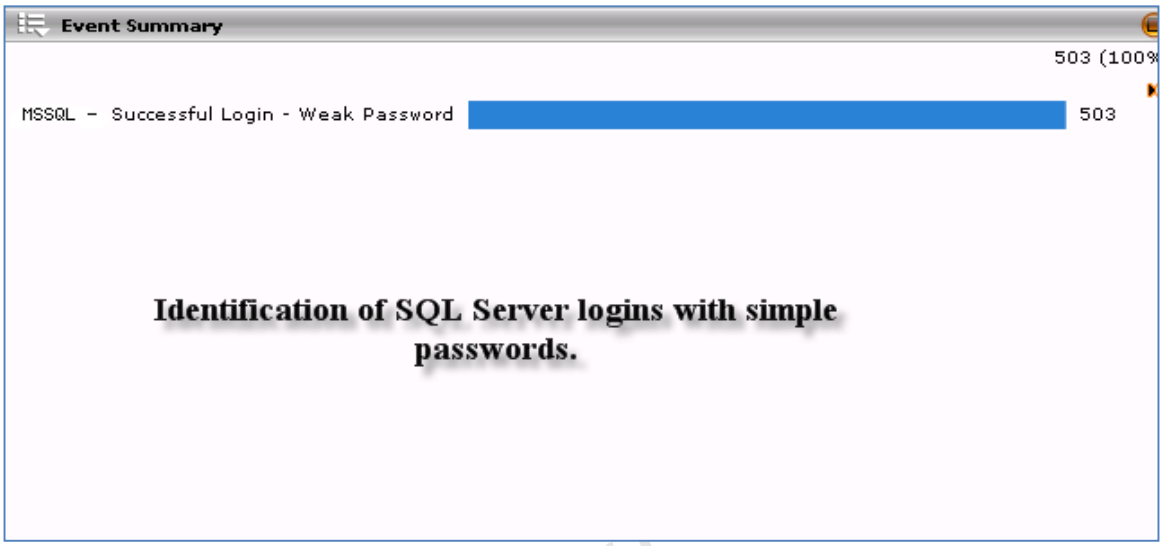
7.2. Weak Authentication

Authentication is critical to any security model, without authentication, it is not possible to assign security roles to accounts. Weak authentication has been a security issues since man started using authentication to protect property. Even though weak authentication as vulnerability has been around a long time, it is something a security program should be able to remediate. Understanding how SQL Server authentication is business critical to a security logging and monitoring project. Tracking user authentications is a basic component of any successful security program. Microsoft SQL Server has two authentication modes: Windows authentication and mixed authentication. Windows authentication relies on Windows to authenticate users and association user with groups using the Kerberos security protocol. SQL Server validates user credentials provided by Windows using Kerberos; the Windows operating system validates the user credentials. SQL Server does not validate the credentials and does not ask for a password. SQL connections classified as trusted are connections made using Windows Authentication because SQL Server trusts the credentials provided by Windows. This method is more secure because it provides password policy enforcement through Active Directory using a corporate security standard. Password behaviors such as complexity, expirations, and lockouts are set and enforced with Active Directory. From a user experience, the user community has a single-sign on and does not need to remember an additional password, or jeopardize the security posture of the environment by using a simple password. Windows Authentication is the default configuration and the recommendation from Microsoft.

When SQL Server uses mixed authentication, users can authenticate either by Windows or with SQL Server. Using mixed authentication SQL Server will authenticate users with Kerberos, when a client connection cannot authenticate using Kerberos, then SQL Server will authenticate the client connection with a username and password stored in SQL Server. SQL Server authentication does not limit or override the permissions of local administrators on the computer where SQL Server resides, the user will retain any elevated rights on the host operating system that are inherent with their account. There is no way to disable Windows Authentication, it is always available and there is no mechanism for disabling it. (Microsoft, 2012)

The recommendation is to use Windows authentication for SQL authentication. This has two benefits for an organization. Corporate password policy will extend to database authentication. This allows user a single-sign on like ability because the Active Directory password will work on the database. If the corporate policy is strong, the database password policy will also be strong. Because of the potentially sensitive nature of database data, protecting access with strong credentials is business critical. Additionally, this will help corporations pass audits.

Most database monitoring appliances have rules for monitor monitoring authentication with simple passwords. Organizations can easily configure reports or alerts when there is an authentication with a simple password. In the table below, there is an example of a database suffering from weak authentication. This is a high-level report, creating a report with the account details is not difficult and will result in giving security staff something actionable.



Report monitoring weak authentication

7.3. SQL Injection

SQL Injection is a common, effective attack that developers can remediate without much cost or lost productivity time. The OWASP organization publishes a SQL injection prevention cheat sheet that will serve as a guide to remediate SQL injection vulnerabilities. The first recommendation is to use stored procedures and prepared statements where possible. When using store procedures, switch them to use parameterized statements instead assembling them on the fly with concatenation, this will reduce the risk of SQL Injection. Below there is a simple example of a stored procedure using dynamic SQL converted to use parameterized SQL.

Before – Vulnerable to SQL Injection

```
ALTER PROCEDURE GetCodes
(
        @ ProductName[nvarchar](32),
        @ProductSkew[nvarchar](32)
)
AS
EXECUTE (`SELECT * FROM Products WHERE ProductName =`"+@ ProductName+`"
AND ProductSkew =`"+@ProductSkew+`")
RETURN
```

After – Parameterized Statement Protected Against SQL Injection

```
SELECT * From Products where ProductName = @ ProductName AND ProductSkew =
@ProductSkew
```

If an organization does not use stored procedures, they should make sure they use prepared statements where possible. Depending on the use, developers can also implement prepared statements. Prepared statements are similar to stored procedures, except the SQL code for a stored procedure id defined and stored in the database, where a prepared statement does not reside in the database. Both of these techniques are effective and organizations should use whichever method best fits their needs.

The next technique to escape user input before placing it in a query. This is a strategy to use when there are concerns about converting dynamic queries into prepared statements and stored procedures. This can be risky because using this technique may cause performance problems in an application. Use this method with caution when there is a need to retrofit legacy code in a cost effective manner. When developing from scratch the best approach is to use parameterized queries.

Implementing the principle of least privilege will minimize the damage from a successful SQL injection attack. As discussed earlier, implementing query-level access control can help minimize the damage from a successful SQL injection attack. The database account used for an application should not run with elevated administrative privileges. If the database account has elevated privileges and there is a successful SQL injection attack, the attacker will own your database and data. Determine what rights service and application accounts need from the beginning, trying to figure out what rights to take away is difficult and can lead to excessive privileges for accounts. Avoid granting create or delete access to database accounts. In cases where accounts require access to portions of data, consider creating a view that limits access to the data in a read-only mode instead of allowing access to the entire table or set of data. Where possible try to limit service accounts access to stored procedures and not to tables themselves. This will be possible if there is a policy to use stored procedures everywhere. (OWASP, 2012)

SQL Server (or any DBMS) should not run as a highly elevated account such as system or root. By default, most databases run under highly privileged account.

SQL Server will run as system by default, a service account with the least privilege should run the DBMS. Running SQL Server with a lower privileged account can minimize the damage if there is an exploit the underlying database. (Ottman, 2010)

Finally, organizations should have a code review for all SQL statements used in an application. During the review process, organizations can verify SQL code conforms to the corporate SDLC standard and best practices. Developers should rewrite SQL code that does not conform to standards bringing it up to a minimum standard.

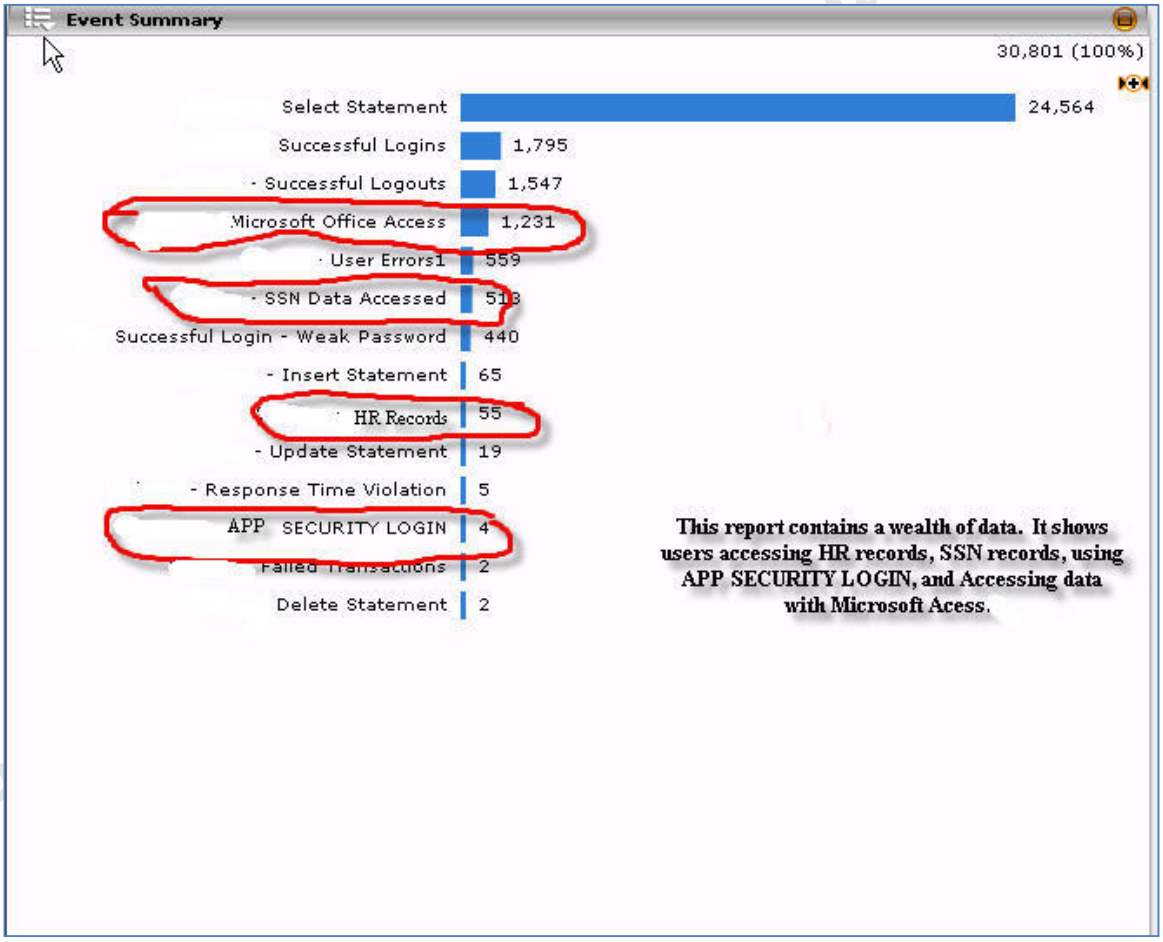
Using a database logging and monitoring appliance to report on SQL injection attempts is easy, all vendors have rules developed to monitor for SQL injection. If the development staff has sloppy coding styles, the SQL injection rules will trigger with false positives. One common trigger is a lazy developer commenting a SQL statement and leaving it in production. This will trigger SQL injection rules. However, if the recommendations above are in place sloppy code should not make into a production environment reducing false positives. (OWASP, 2012)

7.4. Weak Audit Trail

Weak audit trails can be devastating for corporations. In the event of a breach and a corporation cannot tell who accessed what data and how much. Data use and identification is very important and being able to tie users to queries and data access may save a corporation. Identifying sensitive and important data should happen during the design and development process. Once the development team identifies sensitive data, they can take extra precaution for monitoring and logging usage. The security team can use this identification to create custom rules identifying when there is sensitive data access. Being able to indentify data access in a language understandable to business owners will allow security staff to create meaningful reports for business users. A business user getting a report stating there were 27 records accessed does not mean anything to the layperson. However, sending a report that reporting the accessing of 55 HR records gives new meaning to a report. (Shulman, 2006)

The report below is an example of classifying, monitoring and reporting of corporate data. The report shows users accessing HR records, SSN Numbers, using a

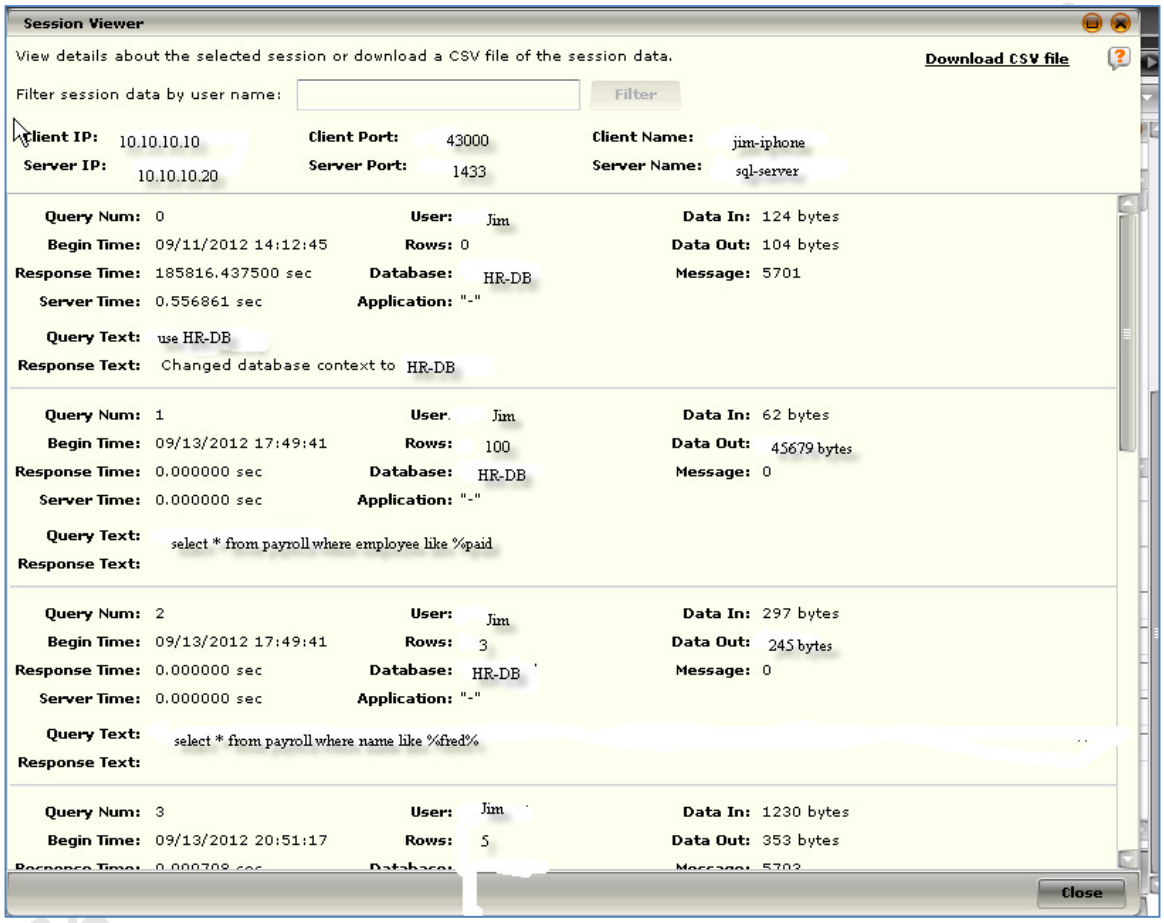
security login, and accessing data with Microsoft Office tools. Using Microsoft Office tools to access data is troubling; the company may have users storing data on laptops, network shares, mobile devices, etc. The example policy in the appendix addresses accessing data with tools such as Microsoft Office. This is a violation of corporate policy triggering an investigation to determine why staff is using Microsoft Office tools to access sensitive data.



SQL Server high-level business-oriented report

In the event there is an event or breach, being able to find out who accessed what data can save a corporation a lot of money. The next slide contains session data from a user "Jim" that access an HR database looking for information. The session data contains all the information needed for forensics and identification of the data accessed by whom,

when and where. This is an example of a strong audit trail that will have many uses for a corporation.

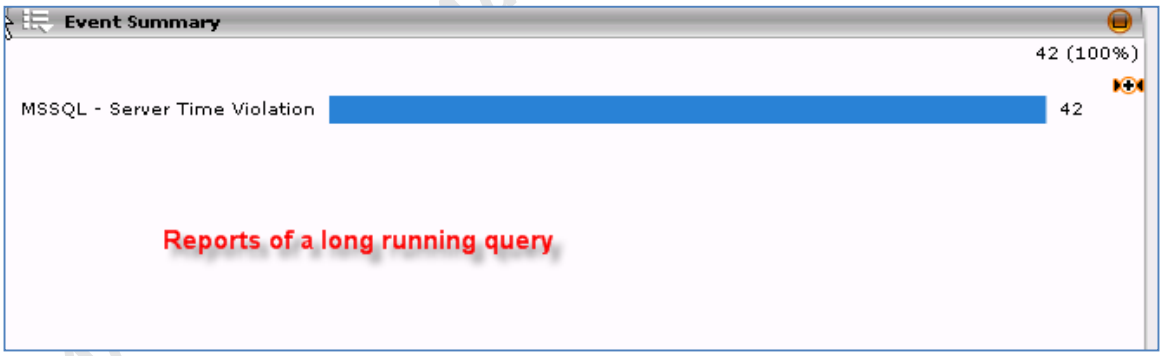


SQL Server database session details

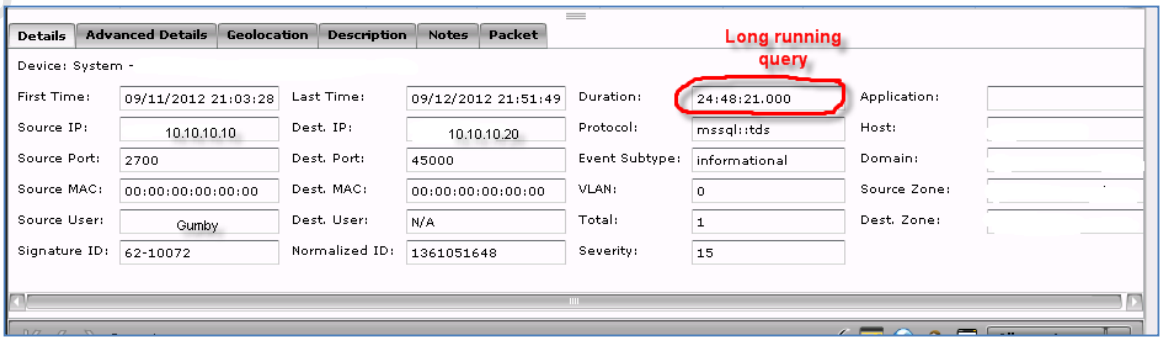
7.5. Database Communication Protocol Vulnerabilities and DoS Attacks

DoS and protocol attacks share the same compensating controls because of their similar nature. Protocol and DoS attacks require a defense in depth approach that should utilize network monitoring, query and response time monitoring, and throttling of traffic if needed. Using a single control to mitigate protocol and DoS attacks will not be effective; there will be too many holes for attackers to exploit. DoS attacks have a goal of service disruption by consuming resources and crashing services. Database protocol attacks usually have nefarious intent such as unauthorized data access, data modification and denial of service. Dealing with database communication protocols is difficult because

protocol attacks will not show up in native database audit logs. Database audit facilities do not cover protocol operations. Implementing a network-based database monitoring system will help detect some, but not all protocol and DoS attacks. Detecting protocol or DoS attacks designed to consume resources is possible with a database-monitoring appliance that is network-based, because the appliance keeps the query responses and captures network connections. A database monitoring solution that leverages network-based monitoring will be able to detect some protocol and DoS attacks to the database. Detecting protocol and DoS attacks is one of the strengths a network-based monitoring and logging solution has over an agent-based solution. An agent-based solution would have difficulty detecting these network-based attacks because the audit logs lack the granularity of the network data. The next screen shot is an example of a simple monitor used for detecting a protocol or DoS attack. (Shulman, 2006)



High-level report of long running queries



Details for one of the long running queries

Investigating the details of a query that ran for 24 hours would be the result of this report. This query could be the result of a normal business process, poorly written query, or a query with malicious intent. On the business end, there is a possibility the query

deals with a large amount of data for month, quarter or year-end. A poorly written query or incorrect database configuration could result in a query running too long. There is also the possibility of an attacker trying to run queries to return large sets of data with the intent of performing a DoS attack. Security personal working with business owners should be able to determine if this query is malicious or part of normal business processes. On a regular basis, organizations should monitor sensitive data access, administrative actions, service and privileged account usage, and suspicious access. These reports will verify employees and processes are following policy, as well as help detect malicious and fraudulent activity.

Limiting monitoring of the environment to only a database appliance is foolish and violates the principle of defense in depth. Hardening of the database and underlying operating system using a best practices guide from an organization such as CIS can also help remove some attack vectors and vulnerabilities. Hardening procedures that target the network can add resiliency to a system when an attacker is probing for vulnerable areas. A word of caution, any procedure used to harden a system should undergo testing to verify the hardening does not affect business operations. Earlier there was a discussion concerning the monitoring of operating system logs looking for security events. Regular monitoring of security events at an operating system is a great way to identify protocol and DoS attacks. As discussed earlier, implementing query-level access control can help minimize the damage from a successful protocol attack. However, query-level access control may not help very much in the case of a DoS attack. SQL Server (or any DBMS) should not run as a highly elevated account such as system or root. By default, most databases run under highly privileged account. SQL Server will run as system by default, a service account with the least privilege should run the DBMS. Running SQL Server with a lower privileged account can minimize the damage if there is an exploit the underlying database. (Natan, 2005)

The recommendation to patch protocol vulnerabilities is obvious. However, the patch lifecycle, change management policies, and business cycles may prevent the deployment of a patch. If there is a vulnerability to a database protocol that runs a critical business function, the patching cycle may need to wait until key business processes

complete. An example of a key business process would be month, quarter or year-end processing.

A very good way to discover DoS and protocol attacks is to view firewall and IDS/IPS (Intrusion Detection System/Intrusion Prevention System) logs. These network-based controls can detect protocol anomalies and potentially stop malicious traffic before it connects to the database. Protocol attacks can be difficult to detect and require the use of several tools to help protect corporate assets. Using one technology will not provide adequate protection of company assets and data.

Finally, if the organization has a SIEM, correlating the data from the network, firewall, IDS/IPS, operating system and database can detect sophisticated attacks that are trying to blend in the rest of the network. SIEM technology is expensive and requires effort and resources to get the maximum benefit, but it can really help with difficult attacks.

7.6. Backup Data Exposure

Regardless of the controls and policies in place, unencrypted backup data leaving the building for an off-site backup, is vulnerable. Imagine implementing all the best practices securing data only to have it fall into malicious hands unencrypted during an off-site transport. A security program is only as strong as its weakest link. Purchasing an appliance for encrypting backups can be expensive, so if the budget is tight, the corporation must undertake an analysis of risk versus exposure m regarding backup media. Implementing an encryption appliance is not trivial; it will require resources to configure the device for the environment. Once the appliance is in place, the device will be subject to routine maintenance and operational responsibility such as patching and hardware repair. Common commercial products do not have built-in encryption for backups available as of this writing. Consumers can hope that at some point, in the future this will become a reality and purchasing expensive backup devices to meet regulatory and business requirements will be available natively in the products.

Purchasing and implementation of encryption devices may depend upon a corporation's appetite for spending, or laws and regulations may mandate it. Monitoring tapes in transit is difficult, there are drive logs and other checks-and-balances, but ultimately a corporation must trust the carrier is doing his/her job. (Shulman, 2006)

8. Conclusion

Implementation of a successful security logging and monitoring program consists of more than deploying a security appliance and turning it on. It is a combination of people, processes and technology. Database security logging and monitoring is more difficult because the data is often sensitive, there are legitimate privileged users or varying degrees, and monitoring requires a risk versus reward versus performance trade-off. Because database monitoring comprises many elements, different pieces of an infrastructure are critical towards the success of the program. Creating a process flow and review for privileged access will address several areas of database security.

The principle of least privilege when implemented correctly can help minimize the damage from manage vulnerabilities. Implementation of query-level access control can help minimize the damage from many successful attacks and vulnerabilities. SQL Server (or any DBMS) should not run as a highly elevated account such as system or root. By default, most databases run under highly privileged account. SQL Server will run as system by default, a service account with the least privilege should run the DBMS. Running SQL Server with a lower privileged account can minimize the damage if there is an exploit the underlying database. These principles should apply to every database instance deployed in the environment, if they are the security posture of the databases increase dramatically.

Using the principle of defense in depth allows organizations multiple tools to help address issues. Leveraging policy, network tools, host tools, and people, is the key to implementing a successful database security logging and monitoring program.

Author Name, email@address

9. References

References

Encryption Hierarchy. (n.d.). *MSDN â€“ Explore Windows, Web, Cloud, and Windows Phone Software Development*. Retrieved July 1, 2012, from http://msdn.microsoft.com/en-us/library/ms189586%28v=sql.100%29

Ashford, W. (2012, July 26). SQL injection attacks rise sharply in second quarter of 2012. *ComputerWeekly.com | Information Technology (IT) News, UK IT Jobs, Industry News*. Retrieved August 1, 2012, from http://www.computerweekly.com/news/2240160266/SQL-injection-attacks-rise-sharply-in-second-quarter-of-2012

Auditing Security Events Best practices: Auditing. (2005, January 25). *Resources and Tools for IT Professionals | TechNet*. Retrieved July 15, 2012, from http://technet.microsoft.com/en-us/library/cc778162%28v=ws.10%29.aspx

Auditing Security Events Best practices: Auditing. (2005, January 25). *Resources and Tools for IT Professionals | TechNet*. Retrieved July 15, 2012, from http://technet.microsoft.com/en-us/library/cc778162%28v=ws.10%29.aspx

B, D. (2012, March 14). â€œThe SQL Guyâ€ Post #20: Using Cell Level Encryption in SQL Server - Canadian IT Professionals - Site Home - TechNet Blogs. *TechNet Blogs - TechNet Blogs*. Retrieved July 1, 2012, from http://blogs.technet.com/b/canitpro/archive/2012/03/14/the-sql-guy-post-20-using-cell-level-encryption-in-sql-server.aspx

Chapman, T. (n.d.). Center for Internet Security :: Security Benchmarks Division :: CIS SQL Server 2005 Benchmark v2.0.0. *Center for Internet Security :: Security Benchmarks Division :: Home*. Retrieved June 27, 2012, from

http://benchmarks.cisecurity.org/en-us/?route=downloads.show.single.sql2005.200

Choose an Authentication Mode. (n.d.). *MSDN*. Retrieved May 20, 2012, from msdn.microsoft.com/en-us/library/ms144284.aspx

Clarke, J. (2009). *SQL injection attacks and defense*. Burlington, MA: Syngress Pub..

Cristofor, L. (2007, October 3). SQL Server 2008: Transparent data encryption feature - a quick overview - Laurentiu Cristofor's blog @microsoft.com - Site Home - MSDN Blogs. *MSDN Blogs - MSDN Blogs*. Retrieved July 2, 2012, from http://blogs.msdn.com/b/lcris/archive/2007/10/03/sql-server-2008-transparent-data-encryption-feature-a-quick-overview.aspx

Kerberos Explained. (n.d.). *Resources and Tools for IT Professionals | TechNet*. Retrieved July 15, 2012, from http://technet.microsoft.com/en-us/library/bb742516.aspx

Kiely, D. (n.d.). Protect Sensitive Data Using Encryption in SQL Server 2005. *SQL Encryption - Microsoft*. Retrieved June 22, 2012, from download.microsoft.com/download/4/7/a/.../SQLEncryption.doc

McKendrick, J. (n.d.). Cost of a data breach: $194 per record | SmartPlanet. *SmartPlanet - Innovative Ideas That Impact Your World*. Retrieved June 8, 2012, from http://www.smartplanet.com/blog/business-brains/cost-of-a-data-breach-194-per-record/22913

McKendrick, J. (n.d.). Cost of a data breach: $194 per record | SmartPlanet. *SmartPlanet - Innovative Ideas That Impact Your World*. Retrieved June 8, 2012, from http://www.smartplanet.com/blog/business-brains/cost-of-a-data-breach-194-per-

record/22913

Northcutt, S. (2010). *Management 404 - Fundamentals of Information Security Policy*. Bethesda, Maryland: The SANS Institute.

SQL Injection Prevention Cheat Sheet. (2012, July 9). *OWASP The Open Web Application Security Project*. Retrieved September 1, 2012, from https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

SQL Injecttion. (2012, July 8). *OWASP The Open Web Application Security Project*. Retrieved September 1, 2012, from https://www.owasp.org/index.php/SQL_Injection

success, a., & category, f. e. (n.d.). Auditing Security Events Best practices: Auditing. *Resources and Tools for IT Professionals | TechNet*. Retrieved July 15, 2012, from http://technet.microsoft.com/en-us/library/cc778162%28v=ws.10%29.aspx

Appendix A

05/20/2012 Database Logging and Monitoring Requirements
Company X
Street Address
City, State XXXX-XXXX
Company X Corporate Security
Policy: Database Logging and Monitoring

Requirements for Database Logging and Monitoring
Auditability is concerned with the features of the application that record activities. Audit trails must provide adequate information to investigate unauthorized activities following a security incident, allowing staff to take the proper remedial action. Company X currently has general standards and guidelines for application auditability. These standards and guidelines referred to as *logging*, should serve as a reference for baseline logging requirements for databases. They are available in the corporate policy under Section 3.4 *Logging*.

Due the nature of Database Management Systems (DBMS) and the likelihood of these systems storing "sensitive" data, this guidance outlines additional requirements for logging to ensure proper auditability and monitoring capabilities. The intention is to have these requirements applied to all production DBMS regardless of the business purpose, and non-production systems that store "sensitive" or "regulated" data. The defined guidelines will require an implementation period for all applicable databases; prior to the transition of any database administration functions to an external third party, the following database audit logging and event monitoring functional controls should be in place.

The following database activity logging should include:

- User Account Additions, Modifications, Suspensions, and Deletions
- User Account changes to Rights (the authorization rights of an account)
- Escalation of privileges
- Object ownership changes
- Login and logout, and failed login attempts of the Administrator Account(s) (account assignment for database administration), Application credentials, and credentials used for direct database access
- Password changes
- Database security policy / configuration changes
    i. Authentication modes
    ii. Password controls
    iii. Remote access enabled or disabled

    iv. Native auditing enabled or disabled
- Audit system configuration changes and attempts to purge, modify, or erase audit trails or database logs
- Sensitive transactions, as required and defined by the data owner
- Allowed access to sensitive resources, as required and defined by the data owner
- Failed access to sensitive resources, as required and defined by the data owner
- Failed SQL attempts to data (object does not exist, insufficient privileges)
- Changes to the database schema (DDL (Data Definition Language) commands)
- Database backup and restore operations
- Database startup and shutdown operation
- Attempts to access OS functionality via the database (execute commands, read / modify files and settings)
- There should be sufficient information in the log record to establish what events occurred and who (or what) caused them:
    i. Type of Event
    ii. When the Event Occurred
    iii. User credential associated with the Event
    iv. Program or Command Used to Initiate the Event (exact SQL)
    v. Names of database tables accessed, if applicable
    vi. Source host name or IP address of the user connection
    vii. Status (success or failure) of the attempt

Platform specific logging:

- ORACLE: log listener commands: SERVICE, STATUS, VERSION, STOP
- MS SQL: DBCC commands

Monitoring should be active for the following logging events:

- User account additions and changes should be reconciled against an account request and approval log
- Significant instances of failed password attempts and against multiple accounts within a short time frame which may indicate hacking attempts
- Significant instances of failed access attempts to the database not authorized to the account ID
- Attempts to SELECT the list of users and passwords
- All direct access to the database from accounts which should be limited to access through an application
- Use of nonstandard tools (E.g. Excel, Access) to directly access DBMS
- Use of any “utility programs” (E.g. Toad) to directly access DBMS
- Use of the Application ID (ApplID) from a source other than the defined owner Application location (based on host name or IP address)
- Log failures, manual logging shut down and attempts to purge
- Attempts to access OS functionality via the database
- Known attack profiles, such as Buffer overflow, Denial of Service, SQL injection

The controls above need assessment and confirmation by the assigned database

custodian. In cases where the database cannot meet the above requirements, corporate security will perform a risk assessment and document the control deficiencies. Corporate security will present this report to Senior Management and stakeholders and request the database custodian sign a risk acceptance form based on the risk assessment performed by corporate security. If database administration needs to transition for a third party, Senior Executive Management must sign a risk acceptance prior to the transition of responsibilities to a third party.

Last Updated: July 26th, 2017

Upcoming SANS App Sec Training

Community SANS Minneapolis DEV534	**Minneapolis, MN**	**Aug 25, 2017 - Aug 28, 2017**	**Community SANS**
Community SANS San Francisco DEV541	**San Francisco, CA**	**Aug 28, 2017 - Aug 31, 2017**	**Community SANS**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
Secure DevOps Summit & Training	**Denver, CO**	**Oct 10, 2017 - Oct 17, 2017**	**Live Event**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017 - DEV522: Defending Web Applications Security Essentials	**Washington, DC**	**Dec 14, 2017 - Dec 19, 2017**	**vLive**
SANS Security East 2018	**New Orleans, LA**	**Jan 08, 2018 - Jan 13, 2018**	**Live Event**
SANS San Francisco Spring 2018	**San Francisco, CA**	**Mar 12, 2018 - Mar 17, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**