

Deploying SQL Server 2008 R2 Based on Payment Card Industry Data Security Standards (PCI DSS) Version 2.0

A White Paper by
Cindy Papas, ParenteBeard LLC
Chad Schieken, ParenteBeard LLC
John Bastow



Contents

Executive Summary	1
1.0 Overview	2
1.1 Payment Card Industry Data Security Standards Background	2
1.2 SQL Server Today	2
1.3 PCI Requirements and SQL Server 2008 R2	2
1.4 Overview of changes in PCI DSS Version 2.0	4
2. SQL Server 2008 R2 Features in Use	6
2.1 Transparent Data Encryption (TDE)	6
2.1.1 Extensible Key Management (EKM)	6
2.1.2 PCI DSS Key Management Requirements	7
2.1.3 Other Encryption Considerations	8
2.2 SQL Server Audit	8
2.3 Enhanced SQL Server 2008 R2 Access Security Features	9
2.3.1 Signed Module	9
2.3.2 Windows Authentication	10
2.3.3 BUILTIN/Administrators Group	10
2.3.4 Role Based Access and Other Access Control Considerations	10
2.4 Default SQL Server 2008 R2 Features	11
2.5 Policy-Based Management (PBM)	11
2.5.1 Encryption Policy	12
2.5.2 SQL Server Audit Policy	13
2.5.3 Role Based Access Policy	13
2.5.4 Authentication Policy	13
2.5.5 Factory Default Settings Policy	14
3.1 Resources	15
3. Conclusion	15

Executive Summary

With technology changes occurring at a rapid pace, IT professionals are continually challenged with the need to balance scalability, compatibility and ease of maintenance to control areas such as security administration and transaction monitoring. Recent data breaches have forced database managers and Chief Information Officers to consider controls to mitigate risks as technology is installed and not as an afterthought. Guidelines from government and industry require a new approach to implementing systems and databases. Organizations that consider compliance as part of their implementation and design stand a better chance of mitigating reputation, security and fraud risk before it can occur.

One of the standards that developers, IT professionals, system architects, and other system administrators of Microsoft SQL Server 2008 R2 should consider as part of their system implementation is the Payment Card Industry Data Security Standards (PCI DSS). Since 2005, more than 252 million credit card records have been breached¹. The increase in data security breaches has prompted and mandated the development of the PCI DSS and other regulations to protect the security of cardholder data. The standards were developed to address the need for strengthening security over sensitive cardholder data elements and providing a foundation for bolstering technical and operational security control activities through a combination of preventive and detective controls and continuous vigilance. The standards consist of twelve data security requirements supporting six control objectives. If you are a company that stores, processes and/or transmits sensitive cardholder data, you are required to demonstrate compliance with PCIDSS.

The key to complying with the standards is to ensure that Information Technology professionals maintain a suitable database platform to allow requirements to be met. Technology professionals need to ensure that they are implemented in an automated and controlled fashion to ensure effective transaction processing. Due to the popularity and proven reputation of SQL Server, critical applications continue to be installed on such highly dependable database software. SQL Server 2008 R2 offers robust new features that provide the means for meeting multiple PCI DSS requirements to properly support key production systems within the organization's environment.

The purpose of this white paper is to provide developers and senior technology leaders with technical solutions on how to proactively achieve PCI compliance when deploying SQL Server 2008 R2 to support and protect key business processes within an organization and avoid the risks noted above.

Note

With the arrival of the PCI DSS 2.0 in October of 2010, and SQL Server R2 we have updated this whitepaper to include coverage of those changes, and other lessons learned since the initial release.

¹ See [Verizon Data Breach Investigations Report](#) for more information on the total number of payment card data breaches

1.0 Overview

1.1 Payment Card Industry Data Security Standards Background

Today, billions of people use payment cards to settle sales transactions. In 2004, five key payment card brands – American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa International – joined together as founding members to establish a common set of information security requirements. The PCI DSS were developed to protect cardholder information and mitigate the risk of loss or theft of payment card related information. The objective of the PCI DSS is to protect cardholder data that is stored, processed or transmitted by merchants, banks, processors, gateway providers, point-of-sale vendors and hardware/software developers.

The most important step to ensuring SQL Server 2008 R2 compliance is to make certain that your understanding of cardholder data and location of such data is clearly understood. “At a minimum, cardholder data contains the full Primary Account Number (PAN). Cardholder data may also appear in the form of the full PAN plus any of the following: Cardholder name, Expiration date, Service code.”²

1.2 SQL Server Today

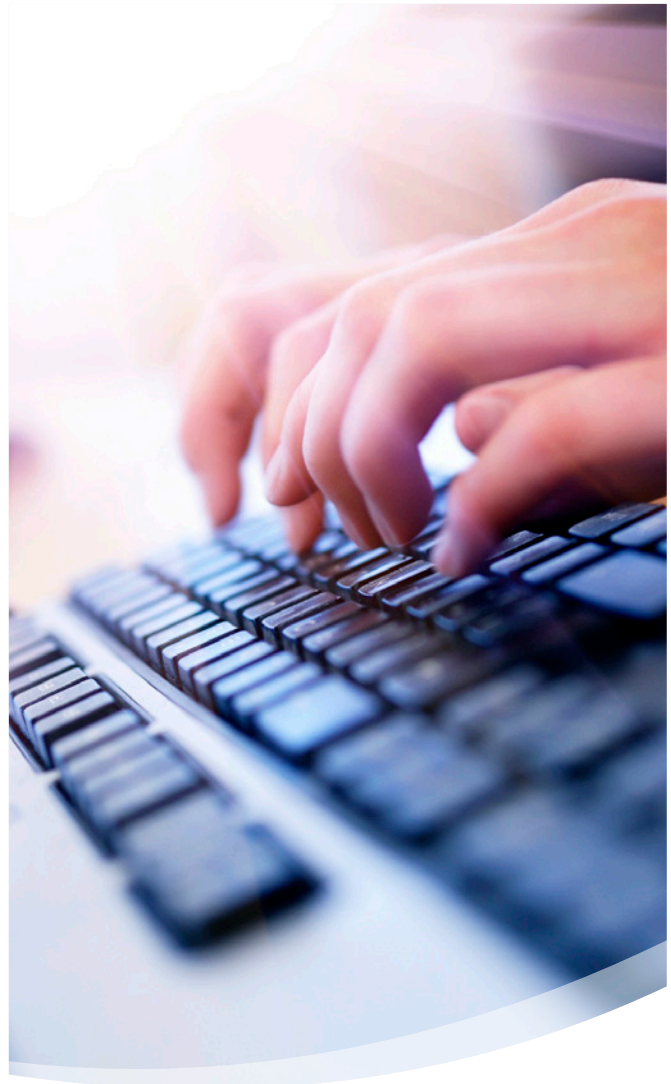
SQL Server has advanced over the years to become a very popular, capable database, evolving from a primarily departmental and SMB database to a fully enterprise capable platform. SQL Server’s appeals are many, from its highly scalable and secure database engine to its built in reporting and data analysis tools. SQL Server 2008 R2 offers new features of particular interest to PCI DSS compliance including:

- Full Database Encryption through Transparent Data Encryption (TDE)
- Split Key Ownership through Extensible Key Management (EKM)
- Granular Auditing Capabilities through SQL Server Audit and Change Data Capture
- Continued support of Signed Module and SSL
- Built-in Control over Default SQL Server 2008 R2 Features
- Stronger Control and Auditability over Server and Database Configuration through Policy-Based Management

Implementation of the PCI DSS controls through SQL Server 2008 R2 technology allows for the ability to standardize and computerize security controls effectively and efficiently, particularly when applied during the installation process.

1.3 PCI Requirements and SQL Server 2008 R2

Six fundamental control objectives are what make up the core information security requirements for PCI DSS version 2.0. In an effort to encourage an information security best practice approach, twelve requirements were established in support of the control objectives. SQL Server 2008 R2 features can be deployed to meet multiple PCI requirements. These requirements should always be considered by Information Technology personnel in cardholder environments when implementing SQL Server 2008 R2.



² www.pcisecuritystandards.org/security_standards/pci_dss_supporting_docs.shtml

The six control objectives and twelve requirements are as follows:

PCI Control Objective: Build and Maintain a Secure Network	
Requirement 1: Install and maintain a firewall configuration to protect cardholder data	N/A - Controlled at the network level
Requirement 2: Do not use vendor-supplied defaults	<ul style="list-style-type: none"> • SQL Server 2008 R2 does not assign default passwords • Most SQL Server 2008 R2 features are disabled by default • The sa account is disabled by default when SQL Server is setup using Windows Authentication • The BUILTIN/Administrators Windows role is not a member of the sysadmin group by default
PCI Control Objective: Protect Cardholder Data	
Requirement 3: Protect stored cardholder data	<ul style="list-style-type: none"> • SQL Server 2008 R2 Transparent Data Encryption offers full data encryption • SQL Server 2008 R2 cell level encryption offers encryption of individual columns • SQL Server 2008 R2 Extensible Key Management offers split encryption key ownership and separation of the keys from the data
Requirement 4: Encrypt transmission of cardholder data across open, public networks	<ul style="list-style-type: none"> • SQL Server 2008 R2 supports Secure Sockets Layer (SSL) encryption
PCI Control Objective: Maintain a Vulnerability Management Program	
Requirement 5: Use and regularly update anti-virus software	<ul style="list-style-type: none"> • N/A - Controlled at the network level
Requirement 6: Develop and maintain secure systems and applications	<ul style="list-style-type: none"> • Change controls are operational in nature; however, segregation of duties is addressed in requirement 7
PCI Control Objective: Implement Strong Access Control Measures	
Requirement 7: Restrict access to cardholder data by business need-to-know	<ul style="list-style-type: none"> • SQL Server 2008 R2 Signed Module facilitates segregation of duties • SQL Server 2008 R2 supports Windows Authentication • SQL Server 2008 R2 supports Role Based Access
Requirement 8: Assign a unique ID to each person with computer access	<ul style="list-style-type: none"> • SQL Server 2008 R2 supports Windows Authentication. Unique identification is preserved even when granted access as a member of a group. • The sa account is disabled by default when SQL Server is setup using Windows Authentication
Requirement 9: Restrict physical access to cardholder data	<ul style="list-style-type: none"> • N/A - Physical access control
PCI Control Objective: Regularly Monitor and Test Networks	
Requirement 10: Track and monitor all access to network resources and cardholder data	<ul style="list-style-type: none"> • SQL Server Audit provides granular auditing capabilities • Once target systems are identified and PCI compliant configurations are set, SQL Policy-Based Management can track changes
Requirement 11: Regularly test security systems and processes	<ul style="list-style-type: none"> • N/A - Controlled at the network level
PCI Control Objective: Maintain an Information Security Policy	
Requirement 12: Maintain a policy that addresses information security	<ul style="list-style-type: none"> • N/A - Operational control procedures

1.4 Overview of changes in PCI DSS Version 2.0

The PCI Security Standards Council has released version 2.0 with the intent of clarifying requirements, and indicating how merchants, service providers, issuers and acquirers should address new or emerging technologies (such as virtualization.) The changes which are most relevant to SQL Server 2008 R2 are outlined below:

Requirement	Change to PCI DDS in 2.0	Impact to SQL Server 2008
General -- Applicability of the PCI DSS	The PCI Security Standards Council has clarified that the cardholder data environment is comprised of "people, processes and technology that store, process, or transmit cardholder data or sensitive authentication data."	Some organizations that may not have determined they needed to comply in the past, now need to comply. Acquiring or issuing banks or financial institutions that use SQL Server 2008 to store, process or transmit cardholder data will need to comply.
2.2.1	Clarified to indicate that if virtualization technology is in use, that each virtual server may have only one primary function.	Any implementation using SQL Server 2008 on a virtual machine should make certain that the underlying virtual windows server is not in use as a web server, DNS server or other component of the CDE.
2.3	Clarified to indicate that administrative access must be via a encrypted transport protocol using strong cryptography.	Users of SQL Server Management Studio will need to make certain the communications between the manager and the SQL server are encrypted using strong cryptography.
3.5	Clarified to explicitly require that any keys used to secure cardholder data must be protected against disclosure and misuse. This now explicitly includes data encryption keys, and key encrypting keys.	<p>Secure storage mechanisms have been defined. Storage mechanisms will either be electronic or manual. One example of a secure electronic storage mechanism would be a cryptographic hardware security module for logical storage.</p> <p>Secure manual storage requires a tamper evident storage container with dual control and split knowledge. Dual control requires two individuals to act in concert to reconstruct a key, and split knowledge requires that each individual have only a piece of the information required to unlock a key for use.</p> <p>Please note that while other methods are permissible, those methods must deliver a similar level of protection.</p>

Requirement	Change to PCI DDS in 2.0	Impact to SQL Server 2008
3.5.1 & 3.5.2	Updated test procedures clarify how key management practices will be evaluated.	Environments not using TDE (our recommended solution) may need to re-evaluate their key management practices in light of the clarifications.
3.6.4	Clarified that keys should be expired “at the end of their crypto period.”	Users of TDE will not need to make changes, while environments with manual key management practices may need to assess their processes to expire keys.
8.5.16	<p>Restricting direct access or queries to databases applies to user access. The statement of the requirement now explicitly states this, whereas before the DSS only included this requirement as a test procedure.</p> <p>Said differently, only database administrators may directly query or access the database.</p>	<p>In our experience, most organizations will not need to make a change in response to this requirement; however all organizations should consider this clarification when assessing their compliance status. The scope of “direct access” in this requirement is intended to indicate who is authorized to submit ad-hoc queries, while network access to do so is addressed by other requirements.</p> <p>Some organizations have found it effective to restrict databases containing cardholder data to only execute stored procedures.</p> <p>Organizations that currently allow ad-hoc queries by power users may need to reconsider the application architecture or implementation. Solutions such as implementing alternative data sources which do not contain cardholder data may be acceptable workarounds; however, there are various tradeoffs which will need to be considered.</p>

2. SQL Server 2008 R2 Features in Use

Several of the exciting new features of SQL Server 2008 enable PCI compliance. These features should aid many organizations to achieve compliance with the PCI DSS when storing sensitive cardholder data in SQL Server databases. For each new feature that is relevant to the PCI DSS, we will present an overview of the feature, provide guidance on its applicability and explain how the feature can be implemented in many common scenarios. In some cases, your configuration requirements may be different than what we have presented, so you should be sure to test your configuration, and consult with the appropriate technical advisors.

2.1 Transparent Data Encryption (TDE)

Limiting sensitive cardholder data storage is critical to minimizing the risk of compromise. We recommend that every environment assess on a periodic basis the need to store cardholder data, including the length of time the data is stored. After completing such an assessment, many organizations will find that they need to continue to store data, for example to process recurring monthly payment transactions.

SQL Server 2008 R2 offers robust data encryption features which may help further reduce the risk of compromise or accidental disclosure. Currently both cell level encryption and TDE are available. We have found that the TDE feature is the most efficient and effective way to meet the PCI DSS requirements for encryption of sensitive cardholder data. TDE encrypts all database files including data files, log files and backup files. In some cases, continued use of cell level encryption may be warranted.

While cell level encryption, which was introduced in 2005, is still supported in SQL Server 2008 R2, many developers have found that cell level encryption does not present a compelling cost/benefit since it requires modifications to client applications in order to handle the explicit encryption and decryption of data. Cell level encryption also prevents the encrypted cells from being indexed or sorted. In particular cases where other requirements dictate specific features not available from either TDE or cell level encryption, some organizations have found that third party add-on products are useful.

"Transparent data encryption" (TDE) performs real-time I/O encryption and decryption of the data and log files. The encryption uses a "Database encryption key" (DEK), which is stored in the database boot record for availability during recovery. The DEK is a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an EKM module. TDE protects data "at rest", meaning the data and log files. It provides the ability to comply with many

laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using AES and 3DES encryption algorithms without changing existing applications."³

When choosing to enable TDE in your environment there are a number of factors to consider during the implementation. First, TDE only secures data at rest and does not help to secure the communication (such as during remote ODBC queries) of the data. Second, the certificate used to encrypt the data encrypting key is required during any attempt to decrypt the data. Third, complete and accurate backups of the certificate are required to minimize the risk of data loss. Backups of the database itself will be encrypted and will require the certificate as well.

2.1.1 Extensible Key Management (EKM)

Using TDE, encryption keys may be protected manually using a database certificate or by utilizing a third-party encryption key management software package through SQL Server 2008 R2's EKM feature. In many cases EKM is the preferred solution, as it allows for external key management and storage without requiring custom programming for many common tasks. EKM with an HSM may not be the appropriate solution for some organizations, based on variety of factors. For those that need an alternative approach to key management, manual key management is available and permissible under the PCI DSS. Please note that manual key management carries with it its own set of security requirements under the PCI DSS 2.0.

EKM is typically used to protect keys used with TDE. To use TDE with EKM, changes are required to the underlying SQL Servers' configuration as well as to the SQL database instance where EKM/TDE will be used. Additionally, an EKM Provider will need to be installed and the SQL Server configured to use the new EKM Provider.

Microsoft has provided several useful overviews which will assist the reader in understanding EKM and enabling TDE using EKM. Please see the following links for further background information:

- [Understanding Extensible Key Management](#)
(Please note that in some circumstances the steps to implement EKM may also be available from the EKM vendor.)
- [How to: Enable TDE Using EKM](#)

³ <http://msdn.microsoft.com/en-us/library/bb934049.aspx>

SQL Server 2008 R2 also allows an organization to use a certificate generated using third party tools. An externally created certificate can be loaded instead of creating a new certificate manually in SQL Server. Once you create the symmetric database encryption key protected by either the certificate or the EKM provided asymmetric key, you can enable TDE to encrypt the data. Please keep in mind that the database encryption key can only be created by SQL Server itself.

2.1.2 PCI DSS Key Management Requirements

PCI DSS addresses the need for data encryption, as well as restricting and protecting access to encryption keys. Keys used for encryption/decryption of cardholder data must be secured against unauthorized use to reduce the risk of data exposure. This can be accomplished through implementation of split key management (e.g., dual control over keys), as well as through the rotation of keys on an annual basis.

Split key management is best handled through an EKM provider. An EKM provider can handle split key management by requiring multiple users to authenticate when performing administrative functions on the keys, such as changing permissions. To ensure segregation of duties, however, please bear in mind that the database owner and/or sysadmin should be independent of the EKM administrator. As another added control feature, EKM also separates the keys from the SQL Server application using the keys, so that keys are not stored with the data.

If you choose not to use an EKM provider you will be required to implement other methods to prevent a single user from having access to both the data and keys outside of the cardholder data environment. First, any user that can backup keys and certificates should have write access to the backup folder location, but be denied read access to that location. Second, users with access to the key and certificate backup folders should be denied access to any backups of the database. To make certain that this is the case, the user who backs up the database should not be the same user who backs up the certificates.

At a high level, if an organization is using manual key management, the key must be stored utilizing tamper evident media, or in a tamper evident container. In some instances something as simple as a pressure-sealed envelope may suffice. The keys must also be placed under dual control. An example of dual control might be a key file an organization has placed in "lockbox" inside of a safe. The key to the lockbox and key to the safe would be given to separate individuals. Thus two people are required to act in concert to recover the key. Lastly,

any plaintext instances of cryptographic keys must be under split knowledge. Split knowledge requires that no single individual has access to the entire plaintext key. Using our "lockbox in a safe" example, split knowledge might require that the actual key is stored in two "halves", and potentially in separate lockboxes inside the safety deposit box.

Looking at what this means for SQL Server when using manual key management first create a database master key in the master database. Be sure to use a strong password to protect the key, with parts of the password entered by two individuals. The database master key you have created will be used to protect the TDE certificate. You are now ready to backup the master database master key and/or the TDE certificate to a removable media. Be certain to store it in a safe location, and employ secure storage mechanisms meeting the requirements of tamper evident, dual knowledge, and split control referenced above. At this point, you are ready to create a certificate in the master database protected by the database master key. Once again, remember to backup the certificate to a removable media and store securely.

When using manual key management, careful consideration must be given to access to the data encrypting keys and key encrypting keys so that your organization can achieve proper implementation of split knowledge. For example, it may be required to have two individuals present to enter portions of the password assigned to the backup certificates. A similar requirement may exist for access to service accounts which can access the keys. Remember to carefully consider which users or service accounts that have sufficient access to the database bootfile, as that will be the "key to the kingdom".

In summary, regardless of whether you are using an EKM provider or manually managing keys, remember that only users who need access to cardholder data should be given permissions to any keys and certificates used to decrypt sensitive data. As noted above, encryption keys may be managed manually or through an EKM provider in SQL Server 2008 R2. In the case of SQL Server, either the EKM generated asymmetric key or certificate that protects the TDE Database Encryption Key must be replaced at the end of its defined crypto period. Generally, the cryptoperiod will be one year; however, as encryption algorithms and computer processing power continue to evolve, this timeframe could lengthen or shorten. You will need to generate or load a new certificate or asymmetric key, backup the certificate, and re-encrypt the Database Encryption Key using the new certificate.

It is important to make sure to keep backups of prior certificates as those will be required to restore copies of the database made when those certificates were active. Keep in mind this is also required when using EKM generated asymmetric keys; however, the EKM provider should have features for managing this.

2.1.3 Other Encryption Considerations

The first way to protect cardholder information from being transmitted across open, public networks is to not send data across open, public networks. If access from database clients is required where public networks are in use, please consider implementing firewalls and a virtual private network (VPN) to protect those communications and limit unauthorized access.

Additionally, data transmitted from a client application of the SQL Server, such as an e-commerce web application, must be encrypted. The most secure SQL Server environment possible will fail if the applications attached to it are not secure. By default, SQL Server transmits data between the server and client unencrypted. SQL Server can be configured to encrypt that data using Secure Sockets Layer (SSL). Although not required by the PCI DSS, the authors recommend always considering the encryption of sensitive cardholder information during transmission, even when data transmission is occurring within a corporate network. In circumstances where such encryption can be achieved without unacceptable tradeoffs, it is prudent to do so.

2.2 SQL Server Audit

The audit functionality in SQL Server 2008 R2 allows for granular control over what is logged. With this feature, actions, tables and users are auditable. In the event of a system compromise, auditing is critical in researching activity associated with a particular scenario. The ability to implicitly log and capture user access to cardholder data, detect changes to database objects/stored procedures, identify changes to server configuration settings and detect modifications to audit configuration settings (e.g., changes to audits and audit specifications) is key to achieving PCI compliance.

When considering what to audit, it is important to balance the need to determine what was accessed and/or manipulated against the amount of data being kept. On a database with high transaction volume, an audit log can rapidly grow to be many times the size of its source database. Keep in mind that an audit trail history must be retained for one year. The guidelines below include minimum data requirement for audit as per the PCI DSS.

First, we recommend that the following system activities be audited or logged:

- **Login attempts** - both successful and failed login attempts
- **Server configuration** - changes to encryption keys, creation, deletion or modification of logins and server level permissions, creation and deletion of databases
- **Database** - creation, deletion or modification of schema objects such as tables, stored procedures and views, addition or deletion of roles and users and changes to their permissions
- **Data** - auditing of any actions, inserts, deletions, updates or selects against tables containing cardholder data

Audits should be configured in such a way as to prevent tampering from SQL Server users, including members of the sysadmin fixed server role and from Windows users who may try to access the audit files without using SQL Server. To accomplish this, we suggest placing audit files in folders or file shares that are not accessible to members of the SQL Server sysadmin role and end users.

Additionally, we suggest giving the SQL Server service account only write access capability to the audit files folder and auditing actions against audits. Examples of actions to audit include changes to audit specifications and enabling or disabling of audits (SQL Server 2008 R2 implicitly does this) and configuring audits to shut down the server if the audit fails. To do this, specify the ON_FAILURE = SHUTDOWN audit option when creating a server audit.

Another option is to write to the Windows Security event log. This allows for increased security over the log data but has a number of potential downsides:

- The security log contains all Windows security log events, not just those for SQL Server, thus making the detection of SQL Server issues more difficult
- Using the event log is slower than writing to a file and can affect server performance
- If the event log is set to overwrite when full, then a malicious user could purposefully fill the event log to cover their tracks

Consider the following to mitigate these risks:

- Send high volume audit information to a file and create another audit to send infrequent but important audit information, e.g., modifying the audit configuration, to the security log
- Frequently send the security log data to a separate secure facility
- Review Security log best practices documented by Microsoft [at this TechNet article](#)

It is recommended that, if using the Windows Security log to record SQL Server audit data, the Audit Collection Services of System Center Operations Manager 2007 be utilized to securely collect and store audit data outside of the log. Second, audit specifications should be defined on the server. These audit groups should be specified on SQL Server for any PCI compliant server as follows:

- SUCCESSFUL_LOGIN_GROUP
- LOGOUT_GROUP
- FAILED_LOGIN_GROUP
- LOGIN_CHANGE_PASSWORD_GROUP
- SERVER_ROLE_MEMBER_CHANGE_GROUP
- BACKUP_RESTORE_GROUP
- DBCC_GROUP
- SERVER_OPERATION_GROUP
- AUDIT_CHANGE_GROUP
- SERVER_STATE_CHANGE_GROUP
- SERVER_OBJECT_CHANGE_GROUP
- SERVER_PRINCIPAL_CHANGE_GROUP
- SERVER_PRINCIPAL_IMPERSONATION_GROUP
- SERVER_OBJECT_OWNERSHIP_CHANGE_GROUP
- SERVER_PERMISSION_CHANGE_GROUP
- SERVER_OBJECT_PERMISSION_CHANGE_GROUP

Database audit specifications should be defined. These audit groups should be applied to any PCI compliant databases and should apply to all users:

- APPLICATION_ROLE_CHANGE_PASSWORD_GROUP
- DATABASE_CHANGE_GROUP
- DATABASE_OWNERSHIP_CHANGE_GROUP
- SCHEMA_OBJECT_CHANGE_GROUP
- DATABASE_PERMISSION_CHANGE_GROUP
- DATABASE_OBJECT_ACCESS_GROUP

The audit groups listed below should be applied to any table in PCI compliant databases and should apply to all users:

- SELECT (Note that SELECT audits capture the SELECT statement and not the resulting data)
- INSERT
- UPDATE
- DELETE

Because the Database Engine can access the audit file, SQL Server logins with CONTROL SERVER permission can use the Database Engine to access the audit files. Define an audit on *master.sys.fn_get_audit_file* to record anyone reading the audit file. This will record which logins

with CONTROL SERVER permission have accessed the audit file through SQL Server.

Although the audit specifications presented above will audit actions, such as inserts updates and deletes, actual changes to data are not audited. To do this, you should enable the Change Data Capture feature against any table containing cardholder data. Change Data Capture creates a change data capture table instance for each table being captured.

The Change Data Capture instance table contains all of the columns in the source table as well as five metadata columns to store information about the change. Unlike trigger based methods for capturing changes, Change Data Capture is implemented asynchronously using the log file and so have a far smaller effect on performance than trigger based methods.

Lastly, it is important to use Windows Authentication or at least an individual SQL Server login for each user as the audit functionality is dependent upon identifying the logged in user in the audit log. If a single application login or shared login is used then identifying a particular user making changes will be impossible. Using Windows authentication may make it easier to link and trace actions beyond the boundaries of the SQL Server.

2.3 Enhanced SQL Server 2008 R2 Access Security Features

Protecting access to cardholder data through effective logical access controls is critical to ensuring PCI requirements are satisfied. Signed modules, introduced in 2005, allows for the facilitation of segregation of duties and continues to be supported in SQL Server 2008 R2. With SQL Server 2008 R2, the sa account is disabled by default when SQL Server is installed using Windows Authentication Mode, the BUILTIN/Administrators group is no longer a member of the sysadmin fixed server role and role based access is supported. These are valuable control features that are discussed in detail below.

2.3.1 Signed Module

A signed module is a trigger, assembly, stored procedure or function that is cryptographically signed by a certificate. The permissions of a certificate user associated with the signing certificate are added to those of the executing user. If the signing user is a member of the sysadmin role, the module can perform any action on the server. Any authorized user of the module runs under that sysadmin context and is granted sysadmin permissions.⁴

⁴ Please see the article [Module Signing \(Database Engine\)](#) for additional details

For example, through a signed stored procedure, a non-sysadmin user could be given the capability of creating new logins and users for the application that they administer without full sysadmin privileges. A good way to accomplish segregation of duties is to encapsulate required sysadmin functionality in signed modules and then grant non-sysadmin users access to those modules.⁵

2.3.2 Windows Authentication

When establishing access management in SQL Server 2008 R2, we recommend using Windows authentication unless client applications require mixed authentication. Additional items for consideration include assigning unique login ID's, avoiding the use of single application login ID's, disallowing users to share logins and creating individually mapped logins instead of creating logins for Windows groups that are members of the sysadmin role. The below login configurations should also be considered:

- Enforce password policy to enforce policy set by local or Active Directory (AD) security policy (this includes both SQL Server logins and, at the AD level, AD logins)
- Enforce 90 day password expiration (this includes both SQL Server logins and, at the AD level, AD logins)
- Set user must change password at next login for new logins
- Assign to database role(s) with the least privileges required
- Restrict access to the database to authorized clients, for PCI DSS 2.0, no "users" may directly query or access the database
- Consider creating an application role in the database to limit what a user can do when connecting directly to a database outside of the application
- SQL Server service accounts should be domain accounts with limited privileges

2.3.3 BUILTIN/Administrators Group

A critical step in restricting access to cardholder data is to limit the number of privileged users assigned to the sysadmin server role. By default, the BUILTIN/Administrators group is not a member of the sysadmin role in SQL Server 2008 R2. The PCI DSS directly supports the best practice concept of "least privilege" access model; therefore, we recommended the following:

- Members of the sysadmin role should only login using individual Windows Logins
- The number of users assigned to the sysadmin role should be minimal
- The sysadmin role should be assigned based on job function
- Members of the sysadmin role's Windows logins should be individually given access to SQL Server rather than through a mapped AD group
- Members of the sysadmin role should not be local Windows administrators
- Members of the sysadmin role should not have access to any folders or file shares on the server that the SQL Server service has access to such as, the folders containing the data and log files, and directories that databases or encryption keys could be backed up to
- A Windows local or domain administrator should not have sysadmin access to SQL Server

2.3.4 Role Based Access and Other Access Control Considerations

The following should also be considered when designing/managing access controls:

- The database owner should not be a sysadmin
- Where SQL Server authentication is allowed, the sa login should be disabled
- SQL Server users should have no local login, RDP or file access to the SQL Server machine
- Database security roles should be setup to restrict access to cardholder data
- Only those users with the need to see cardholder data should be placed in roles with access to cardholder data
- All users, including sysadmin members, should be prevented from being able to modify any audit log files
- Systems developers should not have the ability to make any database modifications to databases on the production server
- Changes to audit specifications should be logged (refer to section 2.2 for additional details)

⁵ Please see SQL Server Separation of Duties Framework on [Codeplex](#) for additional details

2.4 Default SQL Server 2008 R2 Features

One of the most basic ways to defend anything is to decrease its size. This concept is easily applied to a server to ensure that the area is less prone to attack. Reducing surface area on a server is essential to minimizing the risk of a system compromise. In SQL Server 2008 R2 by default, most features, such as database mail and CLR integration, are disabled.

It is important to carefully choose only those items required by your application(s) to be installed or enabled and to monitor the configuration to ensure that unused features continue to be disabled. Each item that is added creates another potential point of attack. To ensure a secure installation, only the database engine itself should be installed. When required add functionality such as Reporting Services, Analysis Services, Integration Services or the Client Tools (see complete list below).

One service that may be enabled by default is the SQL Server Browser. If you are installing only the default instance of SQL Server, the SQL Server Browser service will be disabled. However, when you install a named instance of SQL Server, the SQL Server Browser service is enabled. Keep in mind that the SQL Server Browser service allows client applications to query for the dynamically allocated port of a named instance of SQL Server. By configuring a fixed port for named instances of SQL Server, the SQL Server Browser service is not needed and may be disabled.

SQL Server services which are disabled by default:

- Database Mail
- SQL Mail
- CLR Integration
- OLE Automation
- XPCmdShell
- Ad Hoc Remote Queries
- Web Assistant
- Cross Database Ownership Chaining
- Service Broker
- SOAP

Another way to help increase security during installation or setup is to configure segregation of duties across domain accounts used for SQL Server 2008 R2 and its related services. Configuring services to run using separate accounts prevents a successful compromise of one account leading to the compromise of unrelated data.

When not deploying SQL Server 2008 R2 on Windows Server 2008 or Vista, it is important to properly configure the accounts that SQL Server 2008 R2's various services use. To help ensure that a single compromised account does not impact the entire system, do not use the SYSTEM NETWORK SERVICE or local administrator accounts for any service. In most cases, where these accounts are used, there is no technical basis for doing so.

When deploying SQL Server 2008 R2 on Windows Server 2008 or Vista, the installation process uses the service security identifier (SID) feature of Windows to create unique service SIDs for each service, each of which have their own permissions. This has a similar effect of creating separate accounts for each service even when using the same service account.

Other steps that can be useful in securing the initial setup of the SQL Server include:

- Enable TCP/IP while disabling Named Pipes and VIA (Virtual Interface Architecture)
- On a default instance of SQL Server, change the port number to something other than the default value of 1433. Ensure that client connections can be configured to use a non-default port
- On a named instance of SQL Server, configure a specific port instead of allowing for dynamic ports

2.5 Policy-Based Management (PBM)

Prior to SQL Server 2008 R2, monitoring the configuration of a SQL Server or database would have been a largely manual task, potentially assisted with custom scripts. SQL Server 2008 R2 introduces the concept of PBM. PBM is designed to significantly change the way administrators manage the SQL Server data platform.

Today, database administrators (DBA) spend a large amount of time reacting to issues caused by configuration changes or deployments that do not comply with best practice "standards" or regulatory requirements. PBM allows the DBA to declare the desired state of the SQL Server environment and then manually or automatically check compliance of the system to that desired state.

The DBA declares intent as a Policy. Policy is the unit of automation/action, capturing the desired state (e.g., condition), where to apply (e.g., object set) and when to check (e.g., evaluation mode). The object set can be any entity in the instance, such as databases, tables, views and stored procedures. The evaluation mode can be check on schedule, check on change-log only, check on change-prevent or on demand.

A feature of PBM is that policies can be exported and imported into other servers and, thus, applied across an entire enterprise. In addition, policies can be applied to individual databases. Thus, one set of policies can be applied only to PCI compliant databases while other databases may have more lenient policies applied. There may be a group of policies used to track and monitor access to cardholder data that makes sure there is an audit specification setup for changes to all stored procedures, views and functions.

If a policy evaluation is performed when an object changes, for instance an audit is turned off, it can either log the change or potentially prevent it (in some cases a DBA can decide to override or disable a preventing policy). If a test is performed on a scheduled basis or on demand it simply reports the failure. Target systems should be monitored and periodically validated to ensure that configured PCI relevant security policies are not modified.

2.5.1 Encryption Policy

To continually test and ensure that the settings referenced in section 2.1.1 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Database Facet	@EncryptionEnabled = True	PCI database(s)	
Server Configuration Facet	@ExtensibleKeyManagementEnabled = True	Server	Only if using EKM
Symmetric Key Facet	@EncryptionAlgorithm = TripleDes or AES256	Database Encryption Key	
Certificate Facet	@PrivateKeyEncryptionType = MasterKey	TDE certificate in the Master database	Only if not using EKM
	@KeyLength = 1024	Database Encryption Key	
	@ExpirationDate - @StartDate < 1 Year	TDE certificate in the Master database	Only if not using EKM
	@ExpirationDate > Today's Date	TDE certificate in the Master database	Only if not using EKM

2.5.2 SQL Server Audit Policy

To continually test and ensure that the settings referenced in section 2.2 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Server Facet	@AuditLevel = All	Server	
Audit Facet	@OnFailure = Shutdown	All audits	Be aware that specifying shutdown on failure can potentially cause reliability issues. In particular, avoid writing to network shares due to unreliable network writes.
	@Enabled = True	All audits	
Database Audit Specification	@Enabled = True	All database audit specifications	
Server Audit Specification	@Enabled = True	All server audit specifications	

2.5.3 Role Based Access Policy

To continually test and ensure that the settings referenced in section 2.3.4 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set
Database Security Facet	@IsOwnerSysadmin = False	PCI Database(s)

2.5.4 Authentication Policy

To continually test and ensure that the settings referenced in section 2.3.2 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Server Security Facet	@LoginMode = Integrated	Server	When using only Windows integrated authentication
Login Facet	@LoginType = WindowsUser	Logins that have access to the PCI database(s)	For Windows integrated logins
	@PasswordExpirationEnabled = True	All SQL logins	Only when SQL authentication is enabled.
	@PasswordPolicyEnforced = True	All SQL logins	Only when SQL authentication is enabled.
User Facet	@LoginType = WindowsUser	All users in PCI database(s)	

2.5.5 Factory Default Settings Policy

To continually test and ensure that the settings referenced in section 2.4 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set
Server Facet	@NamedPipesEnabled = False	Server
	@TcpEnabled = True	Server
Server Security Facet	@CrossDBOwnershipChainingEnabled = False	Server
Surface Area Configuration Facet	@AdHocRemoteQueriesEnabled = False	Server
	@ClrIntegrationEnabled = False	Server
	@DatabaseMailEnabled = False	Server
	@OleAutomationEnabled = False	Server
	@ServiceBrokerEndpointActive = False	Server
	@SoapEndpointsEnabled = False	Server
	@SqlMailEnabled = False	Server
	@WebAssistantEnabled = False	Server
	@XPcmdShellEnabled = False	Server
	@ClrIntegrationEnabled = False	Server



3. Conclusion

Implementing Microsoft SQL Server 2008 R2 requires careful planning, analysis and an understanding of the impact to interfaces based on other technology in the environment. System developers and IT leadership can no longer ignore security, privacy and regulatory requirements that mandate compliance with the storage, transmitting and processing of data. This paper highlighted key areas for a developer to focus on to confirm cardholder data is appropriately protected.

Data encryption, dual controls, effective audit trails, accountability, strong password controls, implementation of roles based access and audits of system configuration changes are core security components that are critical to protecting data. Automated implementation of such controls in SQL Server 2008 R2 would allow for the ability to achieve PCI compliance as well as standardize and computerize security controls effectively and efficiently. In order to ensure risk is mitigated in the environment, we encourage developers to continually assess their environments, stay abreast of requirements in the industry and consider the ramifications of not being compliant.

This white paper provided an overview of configurable PCI DSS requirements and proposed solutions, however, the following resources in section 3.1 are provided below for additional reference. We recommend consulting a Qualified Security Assessor (QSA) to evaluate specific configuration issues related to the PCI standards.

3.1 Resources

For more information on implementing SQL Server 2008 R2 based on the PCI Data Security Standards, please visit the following sites:

PCI Security Standards Council website: <https://www.pcisecuritystandards.org>

PCI Data Security Standard version 2.0: http://www.pcisecuritystandards.org/tech/download_the_pci_dss.htm

Navigating the PCI DSS (v2.0) https://www.pcisecuritystandards.org/documents/navigating_dss_v20.pdf

PCI DSS Summary of Changes Version 1.2.1 to 2.0 https://www.pcisecuritystandards.org/documents/pci_dss_v2_summary_of_changes.pdf

PCI Glossary:

http://www.pcisecuritystandards.org/security_standards/pci_dss_supporting_docs.shtml

Understanding Transparent Data Encryption (TDE): <http://msdn.microsoft.com/en-us/library/bb934049.aspx>

Understanding Extensible Key Management (EKM): <http://msdn.microsoft.com/en-us/library/bb895340.aspx>

Understanding SQL Server Audit: <http://msdn.microsoft.com/en-us/library/cc280386.aspx>

SQL Server Audit Action Groups and Actions: <http://msdn.microsoft.com/en-us/library/cc280663.aspx>

Change Data Capture: <http://msdn.microsoft.com/en-us/library/bb522489.aspx>

Module Signing (Database Engine): <http://msdn.microsoft.com/en-us/library/ms345102.aspx>

How to Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager): <http://msdn.microsoft.com/en-us/library/ms177440.aspx>

How to Enable or Disable a Server Network Protocol (SQL Server Configuration Manager): <http://msdn.microsoft.com/en-us/library/ms191294.aspx>

Understanding Surface Area Configuration: <http://msdn.microsoft.com/en-us/library/ms161956.aspx>

Administering Servers by Using Policy-Based Management: <http://msdn.microsoft.com/en-us/library/bb510667.aspx>

Encryption Hierarchy: <http://msdn.microsoft.com/en-us/library/ms189586.aspx>

SQL Server 2008 Compliance Guide <http://www.microsoft.com/downloads/en/details.aspx?FamilyId=6E1021DD-65B9-41C2-8385-438028F5ACC2&displaylang=en>

