Six Ways to Reduce PCI DSS Audit Scope by Tokenizing Cardholder data

nuBridges, inc

nuBRIDGES®

A NUBRIDGES WHITE PAPER

Six Ways to Reduce PCI DSS Audit Scope by Tokenizing Cardholder Data

Enterprises are seeking ways to simplify and reduce the scope of the Payment Card Industry's Data Security Standard (PCI DSS) compliance by shrinking the footprint where cardholder data is located throughout their organization. By reducing the scope, these enterprises can dramatically lower the cost and anxiety of PCI DSS compliance and significantly increase the chance of audit success.

Compliance with the PCI DSS is a combination of documented best practices and technology solutions that protect cardholder data across the enterprise. This paper explores the use of tokenization as a best practice in improving the security of credit card transactions, while at the same time minimizing the cost and complexity of PCI DSS compliance by reducing audit scope.

Introduction

The scope of PCI DSS compliance for any organization is significant both in terms of effort and cost. In a PCI DSS audit, all systems, applications and processes that have access to credit card information, whether encrypted or unencrypted, are considered in scope. The October 2008 update of the PCI DSS documentation (version 1.2) states that companies can reduce the PCI DSS audit scope using network segmentation to isolate the cardholder data in a secure segment. From an application perspective, tokenization functions similarly to network segmentation. These are complementary, not "either/or" approaches for organizations to consider as they map out their data protection and compliance strategies.

The Payment Card Industry Data Security Standard, Version 1.2

"Network segmentation of, or isolating (segmenting), the cardholder data environment from the remainder of the corporate network is not a PCI DSS requirement. However, it is recommended as a method that may reduce:

- The scope of the PCI DSS assessment
- The cost of the PCI DSS assessment
- The cost and difficulty of implementing and maintaining PCI DSS controls
- The risk to an organization (reduced by consolidating cardholder data into fewer, more controlled locations)

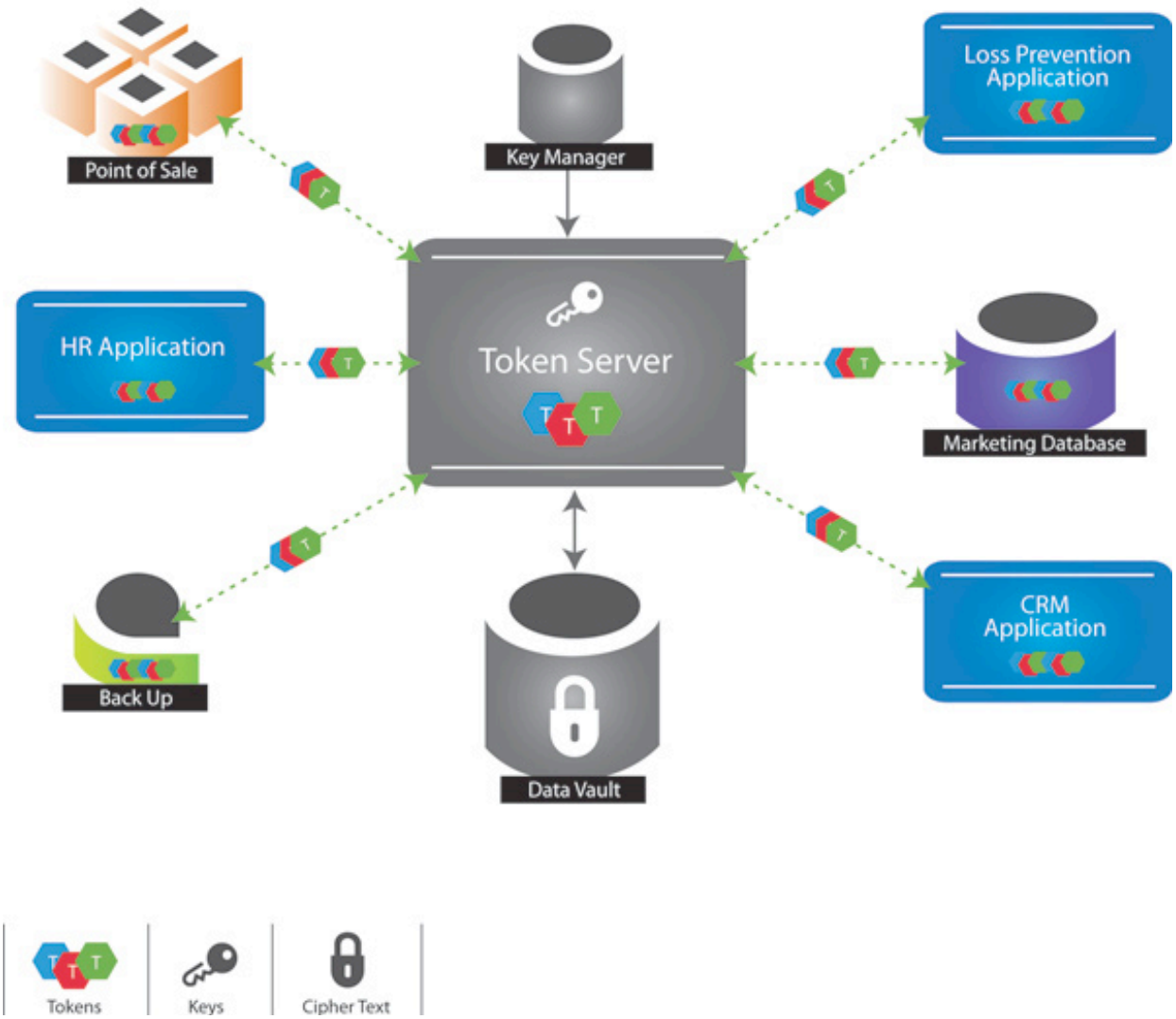
Without adequate network segmentation (sometimes called a flat network), the entire network is in scope of the PCI DSS assessment."

Tokenization Unwrapped

With traditional encryption, when a database or application needs to store sensitive data, those values are encrypted and the resulting cipher text is returned to the original location. With tokenization, a token - or surrogate value - is returned and stored in place of the original data. The token is a reference to the actual cipher text, which is stored in a central data vault. Tokens can be safely used by any file, application, database, or backup medium throughout the organization, minimizing the risk of exposing the actual sensitive data, and allowing business and analytical applications to work without modification.

Organizations that must meet the requirements of PCI DSS are increasingly embracing the compliance benefits of tokenization. Let's take a look at requirement 3.1, which mandates that businesses keep payment data in a minimum number of locations. That is precisely what the tokenization model accomplishes. By using tokenization, businesses are reducing the number of locations where they are retaining cardholder information. Requirements 3.5.1 and 3.5.2 mandate that access to keys be restricted to the fewest number of custodians and that keys be stored securely in the fewest possible locations. With tokenization, encryption is performed centrally when credit card values are tokenized, and keys are centralized on a secure server, optimally addressing these requirements.

Tokenization Model



Six Ways to Reduce PCI DSS Audit Scope

#1: Centralized data vault

Under the tokenization model, encrypted payment card data (called cipher text) is stored only in a central data vault and tokens replace the credit card values in applications or databases. Risk is reduced since credit cards, even in their encrypted state, are not available outside of the data vault, except when originally captured at the beginning of a transaction or, later, accessed from the data vault by authorized and authenticated applications and/or users.

Let's look at how the centralized data value works in a typical tokenization scenario. When a credit card is used for a purchase, the number is transmitted in real-time to the token server. A token representing that data is generated and returned to the calling application and takes the place of the credit card number in that application and all downstream uses of the data (for example, CRM systems, backup copies of the applicaton, exports to data warehouses and exports to fraud detection applications). Meanwhile, the credit card number is encrypted and stored in the central data vault. The clear text value is not stored anywhere. If the need to access the actual value arises (for instance, to process a merchandise return), the value can be decrypted in response to a request by those users or applications with proper authority, and then only for as long as it is needed to complete a transaction. This "round trip" protection limits the risk to the credit card data and ensures that it is never stored in the clear, and that the encrypted cipher-text version of the card data only resides within the secure data vault and in its corresponding back-up/disastaer recover instances. The data vault is never exposed to any applications other than the token server.

#2: Tokens act as data surrogates

With traditional encryption, when an application or database needs to store credit card data, the values are encrypted and cipher text is saved in the original location. With tokenization, a token – or surrogate value – is stored in place of the original data. The token is a reference to the actual cipher text, which is stored in a central data vault.

Encrypted data usually takes up more space than the original values, which requires changes to the applications and/or databases. Tokens can be engineered to preserve the length and format of the original data, so they are almost completely non-invasive to databases and applications; requiring no changes to database schemas, application screens, business processes, etc. This significantly reduces the IT and business impact associated with PCI DSS compliance.

#3: Tokens are surrogates for masked data

Tokens can be generated to preserve parts of the original data values. A typical pattern in PCI DSS scenarios is generating the tokens to maintain the original first two and last four digits of the credit card. A "Token Strategy" provides the flexibility to define the format of the tokens including what part, if any, of the original value to preserve.

Frequently there are applications within the enterprise that need only the last four digits of a credit card to validate credentials. It's this scenario where the use of a format-preserving token allows users to perform their job, validating the last four digits of the credit card. Since the application does not contain credit card information - not even in an encrypted format - the entire application is removed from PCI DSS scope. A word of caution - proper network segmentation is still required.

For example, suppose you call to find out the status of an order you recently placed. The operator, after requesting your name and address, asks you for the last four digits of the credit card you used to make the purchase. He or she verifies what you tell them from the credit card information in their order status application - in this case they see a token, with the last four digits preserved, which appears like a regular credit card. If the data had been stored as cipher text, the application would have had to decrypt in order to present the masked value and it would still be considered part of the PCI DSS audit scope.

#4: One-to-one token/data relationship

Certain types of tokenization can ensure that there is always a one-to-one relationship between the credit card number and the token that is generated so that you maintain referential integrity across multiple systems.

For example, when a retailer needs to understand the buying patterns of consumers they sometimes push transaction data into a data warehouse for analysis. One day a consumer buys a stapler, staples and a notebook using a credit card. A week later that same consumer, using the same credit card, buys staples, paper and notebook tabs. The fact that the consumer bought a stapler followed by staples and a notebook followed by paper is important, and the only way these purchases were linked was through the same credit card number (the credit card number is the "primary key" linking the two).

Maintaining referential integrity allows the data warehouse to perform transaction analysis with tokens, rather than credit card numbers, thus removing it from scope. Once again, network segmentation is also important, but done properly, the scope of the PCI DSS audit is reduced. And the consequences of a breach of the data warehouse have been minimized because any unauthorized acccess to the data warehouse only yields tokens and not actual credit card numbers.

(It should be noted that under certain circumstances localized encryption can be enabled to preserve the one-to-one relationship, generating the same cipher text from the same credit card number. But typically this will limit your ability to introduce randomization into your encryption strategy - in some security scenarios this is not acceptable.)

When you combine #2, #3 and #4, you solve an important security problem for today's IT-dependent enterprises: You eliminate the need to use production data for development and test environments. Often, when application changes are required, testing must be performed across the entire system. Today most enterprises allow developers to use production data, in this case real credit card

numbers, or go through great lengths to obfuscate the data to make it unrecognizable. The same applies to companies that use offshore development labs that don't want to provide production data to them. With the use of tokens, most application testing can take place without having to access real credit card numbers. This reduces the operational effort needed to test applications and can reduce the scope for PCI DSS compliance for applications that only use the format-preserving tokens.

#5: No relationship between tokens and data values

With tokenization there is no mathematical relationship between a token and data value – the only relationship is referential. This is not the case when using encryption or hashing where an algorithm mathematically derives the output cipher text or hash based on the input data. Tokens can be passed around the network between applications, databases and business processes safely, all the while leaving the encrypted data it represents securely stored in a central data vault. Authorized applications that need access to encrypted data can only retrieve it from the tokenization engine, providing an extra layer of protection for credit card data and shrinking the "risk footprint" that needs to be managed and monitored by your security team.

#6: Centralized key management

As mentioned in the introduction, Section 3.5 of the PCI DSS states that keys should be stored "securely in the fewest possible locations and forms." With the use of tokenization, keys are limited to use by the centralized token manager, keeping the distribution of the keys to a minimum. This helps to minimize the scope of PCI DSS compliance and reduce the risk of a key compromise.

Use Case. Cable Company Adopts Tokenization to Reduce PCI DSS Audit Scope

A cable company recently began tokenizing payment card information to reduce the PCI DSS audit process by taking as many systems out of scope as possible. The company stores a bank account number or credit card number for every customer that opts for automatic payment.

Various individuals in the company need access to customer records to perform their jobs, but only the finance department needs to be able to use a bank account or credit card number for monthly billings. For example, a tech support representative needs to be able to see the last four digits of a customer's credit card number to verify the identity of the caller, but not the entire value.

The company adopted a tokenization model whereby the technical support department is a spoke in the hub and spoke model, as are the sales call center, the finance department, business services and installation departments. Using tokenization, where tokens replace sensitive data in customer files, only those employees who need access to confidential customer information have access to cipher text and have the authority to decrypt it. Employees in other departments only see format-preserving tokens in place of cipher text, removing the risk of unauthorized personnel decrypting and using this information.

In this case, the last four digits of each credit card number are preserved so that the tech support department can verify customer identities. The remaining digits are tokenized, and when combined with the preserved last four digits, form tokens that are stored in the application. Unlike cipher text that takes up more room than clear text, in this case tokens take the same amount of room as the clear text credit card value they replace. The use of format-preserving tokens alleviated the need to modify the affected applications.

When a customer wants to change the credit card on file, the tech support representative now simply transfers the customer to the billing department instead of taking the number. Only billing department employees have the authority to record a credit card number. When a replacement credit card number is taken, it is immediately encrypted and stored in the central data vault. The token server then generates a new token for the new credit card number, and returns the token to the billing system for future use as well as to the customer record that is used by other departments. As a result, the applications and systems that contain tokens instead of clear text or encrypted credit card information no longer have to be audited. They are rendered out of scope for PCI DSS compliance.

Introducing a New Tokenization Model

Lightening the load of PCI DSS compliance, while ensuring best practice protection of cardholder information, is at the core of nuBridges' mission. Long a leader in data protection software solutions, nuBridges has taken its offerings to a whole new level by adding nuBridges Protect™ Token Manager to its portfolio. It is the industry's first data protection software solution that combines universal Format Preserving Tokenization™, strong encryption and unified key management in one platform-agnostic package.

About nuBridges, Inc.

nuBridges is a leading provider of software and services to protect sensitive data at rest and in transit, and to transfer data internally or externally with end-to-end security, control and visibility. nuBridges' encryption, key management, managed file transfer and B2B integration solutions are used to comply with security mandates and to digitally integrate business processes among enterprises, systems, applications and people. Over 3,000 customers depend on nuBridges secure eBusiness solutions to encrypt millions of credit cards, exchange billions of dollars in B2B transactions and enable countless business-critical file transfers, including Wal-Mart, Amazon.com, Timberland, American Eagle Outfitters, Costco, Belk, Bon Ton, Wachovia, Sun Trust, AIG, CheckFree and Verizon.

nuBridges is headquartered in Atlanta, Georgia, USA. More information is available at www.nubridges.com.

U.S. HEADQUARTERS
1000 Abernathy Road, Suite 250
Atlanta, GA 30328
p: +1 770 730 3600

EMEA HEADQUARTERS
Lakeside House, 1 Furzeground Way
Stockley Park, Uxbridge
Middlesex, UB11 1BD
United Kingdom
p: +44 (0) 20 8622 3841