A developer's guide to complying with PCI DSS 3.2 Requirement 6

Develop and maintain secure systems and applications

Dave Whitelegg

September 18, 2017
(First published July 09, 2014)

PCI DSS compliance is not about passing an annual audit and ensuring all the right boxes are ticked on the day of the PCI assessment. Every IT system, business process, and staff member in scope of PCI compliance are required to comply all applicable PCI security requirements in a continual state, 365 days of the year. Annual assessments of compliance, even those undertaken by an external PCI Qualified Security Assessor (QSA), validate only the compliance of a limited sample of systems and processes, and in reality provides little assurance on the security state of organization if there is a cardholder data breach. These annual compliance assessments are the reason the payment card industry uses the term "assessment" instead of "audit." There is no official certificate of PCI DSS compliance, as compliance is not judged by at a "moment in time" pass, but ultimately by whether cardholder data is kept secure by meeting all of the requirements all of the time.

What is PCI DSS?

The Payment Card Industry Data Security Standard (PCI DSS) is a highly prescriptive technical standard, which is aimed at the protection of debit and credit card details, which is referred to within the payments industry as cardholder data. The objective of the standard is to prevent payment card fraud, by securing cardholder data within organizations that either accept card payments, or are involved in the handling of cardholder data.

PCI DSS consists of 12 sections of requirements and the responsibility for compliance usually rests with IT infrastructure support. PCI DSS requirement 6 breaks down into 28 further individual requirements and sits squarely with software developers who are involved in the development of applications that process, store, and transmit cardholder data.

PCI compliance heavily revolves around IT services. IT-focused compliance managers that are tasked with achieving compliance within organizations often lack the required software developer knowledge and experience to help assure that the application development meets the arduous
requirements of PCI DSS. Follow along to read a developer's perspective to complying with PCI DSS requirements.

**Understanding the nature of PCI DSS compliance**

PCI DSS compliance is not about passing an annual audit and ensuring all the right boxes are ticked on the day of the PCI assessment. Every IT system, business process, and staff member in scope of PCI compliance are required to comply with all applicable PCI security requirements in a continual state, 365 days of the year. Annual assessments of compliance, even those undertaken by an external PCI Qualified Security Assessor (QSA), validate only the compliance of limited sample of systems and processes. Additionally, the compliance assessments provide little assurance on the security state of an organization when a cardholder data breach occurs; it is the reason the payment card industry uses the term "assessment" instead of "audit."

**There is no official certificate of PCI DSS compliance, as compliance is not judged by a "moment in time" pass. Ultimately, compliance is judged by whether cardholder data is kept secure by meeting all of the requirements, all of the time.**

The objective of a PCI DSS program is to reach and maintain an operational state of compliance with all 300-plus requirements, all of the time. The continual operational goal must always be considered when you are developing processes and controls to satisfy PCI DSS requirements. Anything less than what is required will lead to failure in adequately reducing the risk in avoiding highly negative reputational impact. The costly forensic investigations and significant financial penalties that all come with credit card data theft and fraud, as committed by alarming numbers of ever more cunning and relentless cyber criminals.

Preventing cardholder data breaches is the ultimate goal of PCI DSS and should be the main goal of your PCI DSS program. If your organization’s goal is to "pass" an annual PCI DSS assessment and receive a certificate, then you misunderstand the intent of the standard and will ultimately fail PCI DSS compliance.

From the information security risk perspective, it is important to understand the PCI standard requirements are only concerned with the protection of cardholder data, and the standard sets a minimum level of security requirements to achieve this, based on risk assessment by the Payment Card Industry Security Standard Council (PCI SSC). Although the standard is a good starting point for an information security strategy, PCI DSS is not enough to cover the entire information security piece for any organization. For instance, PCI DSS does not have any requirements that cover availability and system resilience. The PCI SSC is not concerned about payment system resilience, other than ensuring the cardholder data they process and store are kept secure when they do go down. Equally, the minimum controls set out in PCI DSS might not be enough to meet your business' risk appetite. For example, requirement 11.2 states that external vulnerability scans must occur on at least a quarterly basis. For most security-focused organizations, meeting this minimum requirement is not sufficient as the typical best practice and risk assessed outcome is to perform external vulnerability scans on a monthly or even daily basis.
The point is not to take a checklist approach and just follow PCI DSS requirements routine for the sake of following, but to always consider them at a minimum—risk assessing what each requirement addresses, as applying the PCI DSS minimum controls might not be suitable to your organization’s risk appetite. Remember that PCI DSS is only concerned with the protection of cardholder data. Although a highly descriptive and solid security standard, it might not be suitable or a cost effective standard to use to mitigate data theft and loss risk of other confidential data types within your organization.

**PCI development requirements**

Most of the PCI DSS requirements that impact software development fall within requirements 3, 4, and 6 of the standard. Requirements 3 and 4 concern the protection of cardholder data, in that any developed application, which processes, stores, and transmits cardholder data, must meet PCI DSS functional security requirements. These requirements include the statute that developed applications providing necessary access control and account management functionality must meet PCI requirements, such as forcing user password changes every 90 days. These application functionality requirements are typically instructed to the development team from management outside the development function and form an input of application requirements within the software development lifecycle. However, developers are directly responsible for compliance with PCI DSS requirement 6. This section of the PCI standard concerns the development of applications that are used as part of a PCI DSS cardholder data environment, including the development application updates.

**PCI requirement 6: Develop and maintain secure systems and applications**

PCI DSS requirement 6 applies to the development of any internal or external application that is considered as "in scope" of PCI DSS compliance—this refers to any developed application that processes, stores, and transmits cardholder data.

Payment applications that are developed by software vendors for use by multiple external organizations are subject to the Payment Application Data Security Standard (PA-DSS), and are required to be assessed by a PA-QSA (see Related topics).

**Requirement 6.1: Establish a process to identify security vulnerabilities by using reputable outside sources for security vulnerability information and assign a risk ranking to newly discovered security vulnerabilities.**

Compliance with requirement 6.1 requires a documented software asset register of all tools and libraries that are used as part of the software development cycle. Each item within the software asset register will include a version number, how and where software (e.g., development tools and application libraries) is used, and a clear explanation of the function they provide. As software tools and libraries are frequently updated, it is important to have a process to regularly review the software asset register to ensure that the register is up to date.

After establishing a software asset register, implement a process to regularly monitor each item on the register for the notification of security vulnerabilities and updated releases. This monitoring
must be from reputable sources and can be achieved by signing up to official software supplier’s RSS feeds, newsgroups, emailing lists, and security vulnerability news websites (such as Mitre's Common Vulnerabilities and Exposures), which cover the common tools and libraries that are used in software development.

Requirement 6.1 requires a risk ranking to be assigned to all vulnerabilities identified against items within the software asset register. Vulnerabilities are risk assessed and labeled with a risk rating label of "Critical," "High," "Medium," or "Low." A vendor or third-party assessment (such as Mitre) of a risk rating might not be an appropriate or accurate level on how a software tool or library is used by your organization. Therefore, document and include a justification for your organization’s assessed risk level. The risk level helps with the prioritization of patching. Where a patch is not available, assess the urgency and requirements in applying more security controls to mitigate vulnerability risk. Note that any critical vendor security patches must be applied within one month of release as per requirement 6.2.

“PCI Myth Busting: Requirement 6.1 cannot be met by vulnerability scanning.”

The Heartbleed bug (CVE-2014-0160) is a good example of how application vulnerability monitoring works (see Related topics). On April 1, 2014, Google's security team issued a critical vulnerability alert with the OpenSSL library, which is used in millions of the world's web applications to encrypt web network traffic. An organization that adheres to requirement 6.1 would have a software asset register, which included how OpenSSL is used by developed applications, and a process to actively monitor for the notification of OpenSSL vulnerabilities. Upon notification of the OpenSSL Heartbleed vulnerability, it would be clear if and where insecure versions of the OpenSSL library were used by reviewing the software asset register. Given the vulnerability risk rating of Critical, if there were vulnerable versions of the library in use, it would lead to mitigation action—in this case the updating of all vulnerable OpenSSL libraries to known secure versions of the library.

Requirement 6.2: Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release.

Requirement 6.2 builds on the vulnerability monitoring that is covered in 6.1, and requires critical level security patches to be applied within a month of the vendor’s release date. Patches for vulnerabilities that are rated at lower levels must be applied within 2 to 3 months of vendor release. Maintain a log of the patching release monitoring and patching process to ensure patches are quickly identified and applied with the required time frames, and as evidence of compliance with 6.2.

In the Heartbleed bug example, the issue was resolved by installing a version of the OpenSSL library not vulnerable to the bug. Therefore, a month after the OpenSSL critical vulnerability notification, any application that is involved in the processing, transmission, and storage of cardholder data, which was still vulnerable to the Heartbleed bug, cannot be considered as compliant with requirement 6.2, given the availability of non-vulnerable OpenSSL versions.
Considering the criticality of the Heartbleed bug, in reality most organizations sought to apply the patch immediately. Remember, PCI DSS sets a minimum baseline of security requirements, and these minimum levels might not be stringent enough to meet your organization's risk appetite. The noted one-month time period in Requirement 6.2 is an example of this minimal level. For some organizations, a maximum of a month to implement a critical security update was considered too lax a time period, so now there are stronger patching policy requirements. However, where PCI DSS typically differs to industry best practices on information security policies, is that each requirement requires that significant documentation and process logging evidence be kept.

**Requirement 6.3: Develop internal and external software applications (including web-based administrative access to applications) securely.**

Meet these requirements for developing internal and external software applications:

- Be in accordance with PCI DSS (for example, secure authentication and logging).
- Base applications on industry standards and/or best practices.
- Incorporate information security throughout the software-development lifecycle.

Requirement 6.3 requires a formal software development lifecycle, which is based on industry standards or best practices, with information security embedded within each stage of the lifecycle. The software development lifecycle must be documented, specifically detailing how security and PCI requirements are addressed within the definition, design, analysis, and testing phases of development.

The application development documentation must be descriptive in covering how an application processes, transmits, and stores cardholder data. To achieve compliance with 6.3, aim to make the documentation descriptive enough to be understood and followed by a third-party developer alien to your organization.

To ensure and prove that developers adhere to the documented development lifecycle, formally record the completion of each development stage with an independent management review and approval process. And conduct regular audits/reviews of the development process, documenting findings along the way.

“PCI Myth Busting: Requirement 6.3 applies to applications developed by external third parties on behalf of the organization.”

**6.3.1 Remove development, test and/or custom application accounts, user IDs, and passwords before applications become active or are released to customers.**

For compliance with requirement 6.3.1, a process is required to verify all development, test, and customer application accounts. This process needs to be documented as part of the application release cycle, and acts as an audit record that shows the successful completion of an application account review for each formal release or update of the application.
6.3.2 Review custom code before release to production or customers to identify any potential coding vulnerability (by using either manual or automated processes).

Include at least the following when reviewing custom code:

- Have reviewers other than the originating code author to review any code changes. Reviews are individuals who are knowledgeable about code-review techniques and secure coding practices.
- Ensure code is developed according to secure coding guidelines.
- Implement appropriate corrections before release.
- Code-review results are reviewed and approved by management before release.
- Review custom code before release to production or customers to identify any potential coding vulnerability (by using either manual or automated processes).

Requirement 6.3.2 requires a process to review custom code changes by using either manual or automated procedures. While automated tools can review code for common vulnerabilities and mistakes, they should not be relied upon as the sole means for the code review. The code review needs to be sufficient enough to identify code issues, which are difficult or impossible to identify by using automated tools. Therefore, have an individual code reviewer be involved in the code review process. This reviewer must be knowledgeable and experienced in the coding technologies that were used and with secure coding practices, including PCI DSS requirements listed under requirement 6.5. To ensure an objective and independent review, the code review must not be the code author.

As part of the code review, reviewers need to verify adherence to the documented development lifecycle process, per requirements 6.3 and 6.3.1.

The code review policy and process must be documented, and each code review together with findings and outcomes recorded. Management acceptance of each code review prior to each software release should also be documented, and should be part of a formal change control process.
Requirement 6.4: Follow change control processes and procedures for all changes to system components. The processes must include the following requirements:

6.4.1 Separate development/test environments from production environments, and enforce the separation with access controls.

6.4.2 Separation of duties between development/test and production environments

6.4.3 Production data (live PANs) are not used for testing or development

6.4.4 Removal of test data and accounts from system components before the system becomes active / goes into production.

6.4.5 Change control procedures must include the following requirements:

6.4.5.1 Documentation of impact.

6.4.5.2 Documented change approval by authorized parties.

6.4.5.3 Functionality testing so that the change does not adversely impact the security of the system.

6.4.5.4 Back-out procedures.

Under PCI DSS, development and test environments must never be used to store, process, or transmit cardholder data. PCI DSS has a strict set of requirements to ensure that there is a clear separation of application development and test environments from live or production cardholder data environments.

Development and test networks must be isolated from the cardholder data production networks, network segmentation can be achieved through correctly configured firewalls or by using Access Control Lists (ACL) on Switches and Routers. In addition, a separation of staff duties is also required—staff with access and duties with the application development and test environments must not have access and duties to the production environment as well, and vice-versa. Therefore, software developers must not have any access to the production cardholder environment or cardholder data, and designated support staff who support the production environment must not have any access to development and test environments.

The use of cardholder data in development and test environments is never permitted, only dummy test cardholder can be used, which can be either generated or obtained from the card schemes. Test cardholder data is never permitted within production environments.

The application release and deployment must adhere to a formal change control process, including an assessment of impact and functionality testing to verify that changes do not adversely affect security. A backout plan and management approval is also required as part of the change request process.
6.4.6 Upon completion of a significant change, all relevant PCI DSS requirements must be implemented on all new or changed systems and networks, and documentation updated as applicable.

Requirement 6.4.6 is a new addition to the standard and comes into force on January 31, 2018, until then it is considered a best practice. The control requires the PCI DSS requirements to be included within the organizations' change management process, to ensuring compliance controls are considered and not impacted by the change, and PCI DSS documentation, such as network diagrams and asset lists are kept up to date by following changes.

Requirement 6.5: Address common coding vulnerabilities in software development processes as follows.

- Train developers at least annually in up-to-date secure coding techniques, including how to avoid common coding vulnerabilities.
- Develop applications based on secure coding guidelines.

Requirement 6.5 requires developers to be trained in secure coding techniques that are relevant to the application coding languages used. The secure coding techniques must be based on industry best practices or standards and must be documented by the organization to ensure that they are followed by developers. Developer training, which can either be conducted in-house or by a qualified third party, is deemed sufficient for PCI compliance, providing that each developer is able to both identify and resolve all known common coding vulnerabilities, and how sensitive data is handled in memory. Ensuring developers are highly knowledgeable about securing code practices will minimize the number of coding vulnerabilities that are introduced through poor coding practices, saving overall development time and cost.

Developers also need to be aware of the PCI requirements of application handling of cardholder data. For example, the application must never store Sensitive Authentication Data (namely the payment card 4- or 3-digit security code) post payment authorization.

Developers must receive secure code training upon hire and a refresher at least annually. I recommend that developers sign an agreement or pass an exam to demonstrate compliance with requirement 6.5.

Requirements 6.5.1 through 6.5.10 address common application vulnerabilities and coding practices with all types of applications. These controls are based on the latest version of the OWASP Top Ten security risks, and are the minimum controls that must be in place for PCI DSS compliance. The OWASP Top Ten list is regarded as a global industry standard for secure application development. All the PCI coding requirements must be included within software development policy and procedures, and all developers must be assured to be using the secure coding techniques listed in the PCI standard.

For details on meeting the OWASP Top Ten, review Dave Whitelegg's other article, "Scan your app to find and fix OWASP Top 10 2013 vulnerabilities."
Requirement 6.6: For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure that these applications are protected against known attacks by using either of the following methods:

- Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes.
- Installing an automated technical solution that detects and prevents web-based attacks (for example, a web-application firewall) in front of public-facing web applications, to continually check all traffic.

“PCI Myth Busting: Requirement 6.6 cannot be met by vulnerability scanning.”

Applications that are public facing, such as web applications that are accessible over the internet, must be protected by either a Web Application Firewall (WAF) or by a Web Application Vulnerability Scanning process.

There are two methods in which to meet requirement 6.6: Conduct Web Application Vulnerability Scanning or install a WAF. Considering PCI requirements set the minimum level of security controls, some security conscious organizations opt for a "belt and braces" approach with their web application security, and implement both of the 6.6 control methods.

Method 1: Application Vulnerability Security Assessment

Application vulnerability security assessments should be embedded within the development lifecycle, with assessments performed at least annually and after any changes to the application. Automated and/or manual processes, tools, and methods can be used to perform application vulnerability assessments. All detected vulnerabilities must be corrected and the application must be re-evaluated to check whether corrective actions actually address the discovered vulnerabilities.

Not sure if you want to invest in IBM Security AppScan? Download a free trial and try it out for yourself. Read other tutorials on AppScan here.

IBM Security AppScan® is a specialist Web Application Vulnerability Scanning tool that can be used to satisfy PCI DSS requirement 6.6 by using one of the methods. AppScan has pre-set default scanning policies that are able to scan vulnerabilities and report against both PCI DSS Version 3.2 requirements, and the latest OWASP Top Ten best practices. AppScan application assessment reports can be used as evidence of compliance with requirement 6.6.

Method 2: Web Application Firewall

A Web Application Firewall (WAF), which is either a dedicated appliance or a device plug-in, must not be confused with a traditional network layer firewall, which does not inspect traffic at the application layer. Development teams are not responsible for the deployment and management
of WAFs, but WAFs do require application specialist expertise for WAF policies to be effective. For PCI compliance, WAFs must be proven to be effective at preventing common web application vulnerabilities that are listed in requirement 6.5.1 to 6.5.10. Those charged with developing WAF policies must have the relevant WAF expertise, and be in an independent role outside the application development team.

**Requirement 6.7: Ensure that security policies and operational procedures for developing and maintaining secure systems and applications are documented, in use, and known to all affected parties.**

Requirement 6.7 requires the documentation of the SDLC policy and procedures, including explicitly detailing requirements 6.1 through to 6.6. I recommend that you schedule an annual review to ensure all development documentation is kept up to date; it is a good practice to document this process.

Requirement 6.7 also requires all development personnel, and any other staff and third parties that are subject to the development policies and procedures, to be made fully aware of the requirements. Ensure that all documentation is readily available to all personnel involved in the application development process.

For assurance of compliance with 6.7, have all development staff review and sign an acknowledgment of their understanding of the development policy, lifecycle process and security procedures on at least an annual basis.

Finally, ensure any changes to the policies and procedures are always clearly communicated to all staff.

**Maintaining PCI DSS Compliance**

PCI DSS compliance can be thought of as having two phases. The first phase is to reach a PCI DSS compliance state, while the second phase is to maintain a continual PCI DSS state of compliance. The second phase of staying PCI compliant often proves the most difficult to achieve, often due to incorrect assumptions that PCI DSS is about passing an annual assessment, which leads to complacency in maintaining PCI controls and process in between assessments. The secret to successfully maintaining compliance is to build processes that deliver a "business as usual" continued PCI-compliant state. Documenting, keeping records of countless security processes, and implementing management oversight are a necessary evil to prevent complacency from creeping in, and to ensure a PCI DSS compliance state can always be verified and proven at any point in time.
Related topics

- Payment applications that are developed by software vendors for use by multiple external organizations are subject to the Payment Application Data Security Standard (PA-DSS) and are required to be assessed by a PA-QSA.
- Monitoring must be from reputable sources. Sign up for official RSS feeds, newsgroups, emailing lists, and websites like Mitre's Common Vulnerabilities and Exposures (CVE).
- The OpenSSL library was identified as having a critical security vulnerability CVE-2014-0160 referred to as the Heartbleed bug.
- Review the OWASP Top Ten (2013) for further details and guidance.
- Visit the Open Web Application Security Project (OWASP). This open source community formed in 2001 freely produces guidance on application security risks.
- For details on meeting the OWASP Top Ten, review "Scan your app to find and fix OWASP Top 10 2013 vulnerabilities" (Dave Whitelegg, developerWorks, June 2014).
- The Heartbleed bug (CVE-2014-0160) is a good example of how application vulnerability monitoring works.
- Check out a demo web application that is built by IBM for vulnerability testing.
- A Cross-Site Request Forgery (CSRF) forces an user to execute unwanted actions on a web application in which he is authenticated. Read
- Learn more about IBM Security AppScan, a leading web application vulnerability scanning tool.

© Copyright IBM Corporation 2014, 2017
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)