A developer's guide to complying with PCI DSS 3.2.1 Requirement 6

Develop and maintain secure systems and applications

Dave Whitelegg
Published on August 15, 2017

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 for the day of the PCI assessment. Every IT system, business process and staff member in scope of PCI compliance are required comply all applicable PCI security requirements in a continual state of business operation, 365 days of the year. Annual assessments of compliance, even those undertaken by an external PCI Qualified Security Assessor (QSA), only validate compliance of limited sample of systems and processes, and in reality provides little assurance on the security state of organization in the event of a cardholder data breach, and 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 at a ‘moment in time’ pass, but ultimately by whether cardholder data is kept secure by meeting all of the requirements all the time

The objective of PCI DSS programme is to reach and maintain an operational state of compliance with all 300 plus requirements, every second, 24 hours a day, 365 days a year. The continual operational goal must always be considered when introducing processes and controls to satisfy PCI DSS requirements, anything less will lead to failure in adequately reduce the risk in avoiding negative reputational impact, costly forensic investigations, and fines that come with cardholder breaches. After all preventing cardholder data breaches is the sole goal of PCI DSS and should be the ultimate goal of your PCI DSS programme, as opposed to passing an annual PCI assessment.

From the information security and 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 covering data and system availability. The PCI SSC and credit card companies are not concerned about payment system resilience, other than ensuring the cardholder data they process and store is kept secure when they do go down. Equally, the minimum controls set out in PCI DSS may not be enough to meet your business's risk appetite. For example, requirement 11.2 states external vulnerability scans must occur on at least a quarterly basis. For most 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 tick box approach and follow PCI DSS requirements robotically, but instead to always consider the security risk that each requirement addresses, as applying the PCI DSS minimum controls might not be suitable to your organization risk appetite. Remember PCI DSS is only concerned with the protection of cardholder data, although a highly descriptive and solid security standard, may 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 PCI DSS requirements that affect 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, this includes elements such as access control and account password management. Therefore any internally developed application control requirements required to meet PCI DSS functional requirements, such as forcing user password changes every 90 days, need to be instructed to the developer team as part of software development lifecycle, as application functionality requirements. Developers are directly responsible for compliance with PCI DSS requirement 6, this section of the PCI standard concerns the internal development of applications in scope of PCI DSS, and this also includes the development of 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 means 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).

6.1 Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as “high,” “medium,” or “low”) to newly discovered security vulnerabilities.

Compliance with requirement 6.1 starts by identifying and documenting a software asset list of all tools and libraries that are used within the development process, including those assets deployed with applications. The software asset list must detail a version, where tools and libraries are used, and an explanation of the function they provide. As software tools and libraries are frequently changed by developers, it is important to regularly review the software asset list to ensure that it is always kept up to date.

After you establish a software asset list, implement a process to regularly monitor each item on the list for notification of vulnerabilities and patches. Monitoring must be from reputable sources and can be achieved by signing up to official RSS feeds, newsgroups, emailing lists, and websites like Mitre's CVE), which covers most of the common tools that are used in development. Be aware that more niche development tools and libraries, especially those directly sourced from a third party, require direct monitoring of the software vendor and open source websites.

Upon discovery of a new vulnerability for a tool or library that is listed on the software asset list, it first must be risk that is assessed and labeled with a risk rating (High, Medium, or Low). This PCI requirement allows the prioritization of patching, or if a patch is not available, helps the decision making of whether to apply more security controls to mitigate vulnerability risk.

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 ideally works (see Related topics). On April 1, 2014, Google's security team issued a critical vulnerability alert with the OpenSSL library, a library that is used in over 500,000 of the world's web applications to encrypt web traffic. An organization compliant with version 6.1 would already understand its usage of OpenSSL within the cardholder environment, and therefore would actively monitor for notifications of vulnerabilities and patches with regard to OpenSSL. Therefore, after Google's alert, the development team would have risk assessed the vulnerability and applied a risk rating.

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. When a critical vulnerability becomes known and a patch is available to resolve it, the patch must be applied within a month. Patches for vulnerabilities rated at lower levels of criticality must be applied within 2 to 3 months. Maintain a patching audit log as evidence of compliance.

In the Heartbleed bug example, the issue was resolved by installing a version of the OpenSSL library that is 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 PCI DSS compliant.

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 rigid enough to meet your organization's risk appetite. The one month time period listed in Requirement 6.2 is an example of this minimal level. Many organizations consider it to be a too lax a time period for the application of critical patches.

6.3 Develop internal and external software applications (including web-based administrative access to applications) securely, as follows:

- In accordance with PCI DSS (for example, secure authentication and logging)
- Based on industry standards and/or best practices.
- Incorporating information security throughout the software-development life cycle

Requirement 6.3 helps ensure that applications are developed in a structured manner, by using development processes that adhere to known secure best practices.

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

Applications must be developed in accordance with a formal software development lifecycle, which is based on industry standards or best practices, with 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.

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 that all developer accounts are removed from released applications. This process needs to be documented as part of the application release cycle, including an audit record that shows the successful completion of an account review for each release of an application by development.

6.3.2 Review custom code prior to release to production or customers in order to identify any potential coding vulnerability (using either manual or automated processes) to include at least the following:

- Code changes are reviewed by individuals other than the originating code author, and by individuals knowledgeable about code-review techniques and secure coding practices.

- Code reviews ensure code is developed according to secure coding guidelines
- Appropriate corrections are implemented prior to release.
- Code-review results are reviewed and approved by management prior to release.
- Review custom code prior to release to production or customers to identify any potential coding vulnerability (using either manual or automated processes) to include at least the following.

Requirement 6.3.2 requires a code review to be conducted, which must be completed by individuals who are not the code author.

The code reviewers must be knowledgeable and experienced in both the coding technologies that are used, and with secure coding practices, including PCI DSS requirements that are listed under requirement 6.5.

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 process and findings must be documented and recorded. Management must approve the completion of the code review before any application release.

6.4 Follow change control processes and procedures for all changes to system components. The processes must include the following:

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:

6.4.5.1 Documentation of impact.

6.4.5.2 Documented change approval by authorized parties.

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

6.4.5.4 Back-out procedures.

PCI DSS does not regard development and test environments as ever secure enough environments to contain cardholder data. Therefore, the standard has a strict set of requirements to ensure the separation of development and test environments from production environments.

Development and test environments must be network segmented from the rest of the cardholder data network. Use a firewall or Access Control Lists (ACL) on a Switch to achieve this separation. A separation of duties with staff is also essential for compliance. Staff with access to the development and test environments must not have any access to the production environment, and vice-versa.

The use of any cardholder data in development and test environments is never permitted, nor is test cardholder data ever permitted within production environments.

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.

All changes to the application must adhere to a change control process, including an assessment of impact and functionality testing to verify that changes do not adversely affect security. A backout plan and formal management approval is also required for each change request.

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 once each developer is able to both identify and resolve all known common coding vulnerabilities.

Developers must be instructed to code applications to handle cardholder data in a PCI DSS-compliant manner. As an example, never store Sensitive Authentication Data (SAD), namely the payment card 4 or 3 digit security code post payment authorization.

Developers should receive secure code training upon hire and at least annually. It is recommended to have developers sign an agreement or ideally pass an exam to demonstrate compliance with requirement 6.5. *Requirements 6.5.1 through 6.5.10 are directly based on the current OWASP Top Ten guidance. The OWASP Top Ten is regarded as industry best practice for secure application development. Requirements 6.5.1 to 6.5.10 are subject to change outside the PCI DSS three-year update cycle. When OWASP releases an updated top ten, be aware that PCI DSS requirements 6.5.1 to 6.5.10 instantly changes to reflect the latest best practice.

For details on meeting the OWASP Top Ten, review the article, "Scan your app to find and fix OWASP Top 10 vulnerabilities." If the OWASP Top Ten is not a relevant security standard for the application developed, an alternative best practice or standard must be referenced instead, which addresses common coding vulnerabilities in the application technology used.

6.6 For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by 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

(Web) Applications that are (public) Internet facing, must be protected by either a Web Application Firewall (WAF) or by Web Application Vulnerability Scanning process.

Requirement 6.6 gives a binary choice of either conducting web application vulnerability scanning, or installing a web application firewall. Considering PCI requirements set the minimum level of security controls, it is common for organizations to opt for a "belt and braces" approach with their web application security, and adopt both of the 6.6 controls.

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 independent from the development team.

Web application vulnerability scanning should be part of the development lifecycle, and must occur at least annually, and after any changes in the application. Scanning can be manual or be a specialist tool led process, which as a minimum, must test the application for all the vulnerabilities that are listed under requirement 6.5. Where vulnerability scanning discovers application vulnerabilities, they must be mitigated and the web application rescanned, as only a clean scan report is acceptable as evidence of compliance.

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

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.

The documentation of software development policies and procedures are required as requirements 6.1 through to 6.6. Requirement 6.7 requires these policies and procedures to be reviewed and maintained regularly. It is recommended to schedule an annual review to ensure all development documentation in requirement 6 is kept up-to-date; this process itself should also be documented.

Requirement 6.7 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. Therefore, all documentation must be readily available to all personnel involved in application development, for instance, allowing staff to find and access the documentation by placing them onto the company intranet.

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

Finally, ensure any changes that are made 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, due to organizations not understanding PCI DSS is not really about passing an annual, and general complacency to maintaining PCI related process in-between assessments. The secret to successfully maintaining compliance is to build processes deliver a "business as usual" continued PCI-compliant state. Documenting, keeping records of the security processes and strong management oversight are a necessary evil to prevent complacency from creeping in, and to ensure PCI DSS compliance 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 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 vulnerabilities (Dave Whitelegg, developerWorks, June 2014).
- The Heartbleed bug (CVE-2014-0160) is a good example of how application vulnerability monitoring should work.
- 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.
- Visit the Security on developerWorks community to find more how-to-guides, articles, videos, and demos in our community resource library.
- Visit the Security on developerWorks blog to learn about new security-related how-to guides, articles, and demo videos.
- Sign up for the Security on developerWorks newsletter for the latest security headlines.
- Follow @dwsecurity to get updates from the developerWorks security zone in real time.
- Learn more about IBM Security AppScan, a leading web application vulnerability scanning tool.