Requirement 6: Develop and maintain secure systems and applications

Unscrupulous individuals use security vulnerabilities to gain privileged access to systems. Many of these vulnerabilities are fixed by vendor-provided security patches, which must be installed by the entities that manage the systems. All critical systems must have the most recently released, appropriate software patches to protect against exploitation and compromise of cardholder data by malicious individuals and malicious software.

Note: Appropriate software patches are those patches that have been evaluated and tested sufficiently to determine that the patches do not conflict with existing security configurations. For in-house developed applications, numerous vulnerabilities can be avoided by using standard system development processes and secure coding techniques.

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
6.1 Ensure that all system components and software are protected from known vulnerabilities by having the latest vendor-supplied security patches installed. Install critical security patches within one month of release. **Note:** An organization may consider applying a risk-based approach to prioritize their patch installations. For example, by prioritizing critical infrastructure (for example, public-facing devices and systems, databases) higher than less-critical internal devices, to ensure high-priority systems and devices are addressed within one month, and addressing less critical devices and systems within three months.	**6.1.a** For a sample of system components and related software, compare the list of security patches installed on each system to the most recent vendor security patch list, to verify that current vendor patches are installed.			
	6.1.b Examine policies related to security patch installation to verify they require installation of all critical new security patches within one month.			

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
6.2 Establish a process to identify and assign a risk ranking to newly discovered security vulnerabilities. **Notes:** ▪ Risk rankings should be based on industry best practices. For example, *criteria for ranking —High‖ risk* vulnerabilities may include a CVSS base score of 4.0 or above, and/or a vendor-supplied patch classified by *the vendor as —critical,‖ and/or a* vulnerability affecting a critical system component. ▪ The ranking of vulnerabilities as defined in 6.2.a is considered a best practice until June 30, 2012, after which it becomes a requirement.	**6.2.a** Interview responsible personnel to verify that processes are implemented to identify new security vulnerabilities, and that a risk ranking is assigned to such vulnerabilities. (At minimum, the most critical, highest risk vulnerabilities should be ranked as —High.‖			
	6.2.b Verify that processes to identify new security vulnerabilities include using outside sources for security vulnerability information.			
6.3 Develop software applications (internal and external, and including web-based administrative access to applications) in accordance with PCI DSS (for example, secure authentication and logging), and based on industry best practices. Incorporate information security throughout the software development life cycle. These processes must include the following:	**6.3.a** Obtain and examine written software development processes to verify that the processes are based on industry standards and/or best practices.			
	6.3.b Examine written software development processes to verify that information security is included throughout the life cycle.			
	6.3.c Examine written software development processes to verify that software applications are developed in accordance with PCI DSS.			
	6.3.d From an examination of written software development processes, and interviews of software developers, verify that:			
6.3.1 Removal of custom application accounts, user IDs, and passwords before applications become active or are released to customers	**6.3.1** Custom application accounts, user IDs and/or passwords are removed before system goes into production or is released to customers.			

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
6.3.2 Review of custom code prior to release to production or customers in order to identify any potential coding vulnerability. **Note:** This requirement for code reviews applies to all custom code (both internal and public-facing), as part of the system development life cycle. Code reviews can be conducted by knowledgeable internal personnel or third parties. Web applications are also subject to additional controls, if they are public facing, to address ongoing threats and vulnerabilities after implementation, as defined at PCI DSS Requirement 6.6.	**6.3.2.a** Obtain and review policies to confirm that all custom application code changes must be reviewed (using either manual or automated processes) as follows: ▪ Code changes are reviewed by individuals other than the originating code author, and by individuals who are knowledgeable in code review techniques and secure coding practices. ▪ Code reviews ensure code is developed according to secure coding guidelines (see PCI DSS Requirement 6.5). ▪ Appropriate corrections are implemented prior to release. ▪ Code review results are reviewed and approved by management prior to release.			
	6.3.2.b Select a sample of recent custom application changes and verify that custom application code is reviewed according to 6.3.2.a, above.			
6.4 Follow change control processes and procedures for all changes to system components. The processes must include the following:	**6.4** From an examination of change control processes, interviews with system and network administrators, and examination of relevant data (network configuration documentation, production and test data, etc.), verify the following:	▨	▨	▨
6.4.1 Separate development/test and production environments	**6.4.1** The development/test environments are separate from the production environment, with access control in place to enforce the separation.			
6.4.2 Separation of duties between development/test and production environments	**6.4.2** There is a separation of duties between personnel assigned to the development/test environments and those assigned to the production environment.			
6.4.3 Production data (live PANs) are not used for testing or development	**6.4.3** Production data (live PANs) are not used for testing or development.			
6.4.4 Removal of test data and accounts before production systems become active	**6.4.4** Test data and accounts are removed before a production system becomes active.			

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
6.4.5 Change control procedures for the implementation of security patches and software modifications. Procedures must include the following:	**6.4.5.a** Verify that change-control procedures related to implementing security patches and software modifications are documented and require items 6.4.5.1 – 6.4.5.4 below.			
	6.4.5.b For a sample of system components and recent changes/security patches, trace those changes back to related change control documentation. For each change examined, perform the following:	▨	▨	▨
6.4.5.1 Documentation of impact.	**6.4.5.1** Verify that documentation of impact is included in the change control documentation for each sampled change.			
6.4.5.2 Documented change approval by authorized parties.	**6.4.5.2** Verify that documented approval by authorized parties is present for each sampled change.			
6.4.5.3 Functionality testing to verify that the change does not adversely impact the security of the system.	**6.4.5.3.a** For each sampled change, verify that functionality testing is performed to verify that the change does not adversely impact the security of the system.			
	6.4.5.3.b For custom code changes, verify that all updates are tested for compliance with PCI DSS Requirement 6.5 before being deployed into production.			
6.4.5.4 Back-out procedures.	**6.4.5.4** Verify that back-out procedures are prepared for each sampled change.			
6.5 Develop applications based on secure coding guidelines. Prevent common coding vulnerabilities in software development processes, to include the following:	**6.5.a** Obtain and review software development processes. Verify that processes require training in secure coding techniques for developers, based on industry best practices and guidance.			
	6.5.b Interview a sample of developers and obtain evidence that they are knowledgeable in secure coding techniques.			
Note: The vulnerabilities listed at 6.5.1 through 6.5.9 were current with industry best practices when this version of PCI DSS was published. However, as industry best practices for vulnerability management are updated (for example, the OWASP Guide, SANS CWE Top 25, CERT Secure Coding, etc.), the current best practices must be used for these requirements.	**6.5.c.** Verify that processes are in place to ensure that applications are not vulnerable to, at a minimum, the following:	▨	▨	▨

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
6.5.1 Injection flaws, particularly SQL injection. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws.	**6.5.1** Injection flaws, particularly SQL injection. (Validate input to verify user data cannot modify meaning of commands and queries, utilize parameterized queries, etc.)			
6.5.2 Buffer overflow	**6.5.2** Buffer overflow (Validate buffer boundaries and truncate input strings.)			
6.5.3 Insecure cryptographic storage	**6.5.3** Insecure cryptographic storage (Prevent cryptographic flaws)			
6.5.4 Insecure communications	**6.5.4** Insecure communications (Properly encrypt all authenticated and sensitive communications)			
6.5.5 Improper error handling	**6.5.5** Improper error handling (Do not leak information via error messages)			
6.5.6 All ―High‖ vulnerabilities identified in the vulnerability identification process (as defined in PCI DSS Requirement 6.2). **Note:** This requirement is considered a best practice until June 30, 2012, after which it becomes a requirement.	**6.5.6** All ―High‖ vulnerabilities as identified in PCI DSS Requirement 6.2.			
Note: Requirements 6.5.7 through 6.5.9, below, apply to web applications and application interfaces (internal or external):				
6.5.7 Cross-site scripting (XSS)	**6.5.7** Cross-site scripting (XSS) (Validate all parameters before inclusion, utilize context-sensitive escaping, etc.)			
6.5.8 Improper Access Control (such as insecure direct object references, failure to restrict URL access, and directory traversal)	**6.5.8** Improper Access Control, such as insecure direct object references, failure to restrict URL access, and directory traversal (Properly authenticate users and sanitize input. Do not expose internal object references to users.)			
6.5.9 Cross-site request forgery (CSRF)	**6.5.9** Cross-site request forgery (CSRF). (Do not reply on authorization credentials and tokens automatically submitted by browsers.)			

PCI DSS Requirements	Testing Procedures	In Place	Not in Place	Target Date/ Comments
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 a web-application firewall in front of public-facing web applications	**6.6** For public-facing web applications, ensure that either one of the following methods are in place as follows: ▪ Verify that public-facing web applications are reviewed (using either manual or automated vulnerability security assessment tools or methods), as follows: – At least annually – After any changes – By an organization that specializes in application security – That all vulnerabilities are corrected – That the application is re-evaluated after the corrections ▪ Verify that a web-application firewall is in place in front of public-facing web applications to detect and prevent web-based attacks. **Note:** *—An organization that specializes in application security‖ can be either a third*-party company or an internal organization, as long as the reviewers specialize in application security and can demonstrate independence from the development team.			