

Requirement 6: Secure Development (6.3 6.5 6.6 6.7)



John Elliott

PRIVACY, PAYMENTS, SECURITY AND RISK SPECIALIST

@withoutfire www.withoutfire.com





Requirement 6.3

Develop software applications securely





Requirement 6.3

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

In accordance with PCI DSS

Based on industry **standards** and/or **best practices**.

Incorporating information **security** throughout the **software-development life cycle**



Requirement Guidance

- Without the inclusion of security during the requirements definition, design, analysis, and testing phases of software development, security vulnerabilities can be inadvertently or maliciously introduced into the production environment.
- Understanding how sensitive data is handled by the application—including when stored, transmitted, and when in memory—can help identify where data needs to be protected.

Beware of These Words

... In accordance
with PCI DSS ...



Hang on, PCI DSS
doesn't have any
software
development
standards



... In accordance
with PCI DSS ...

Storing and encrypting PANs (3)

Encryption in transit (4)

Vulnerability management (6)

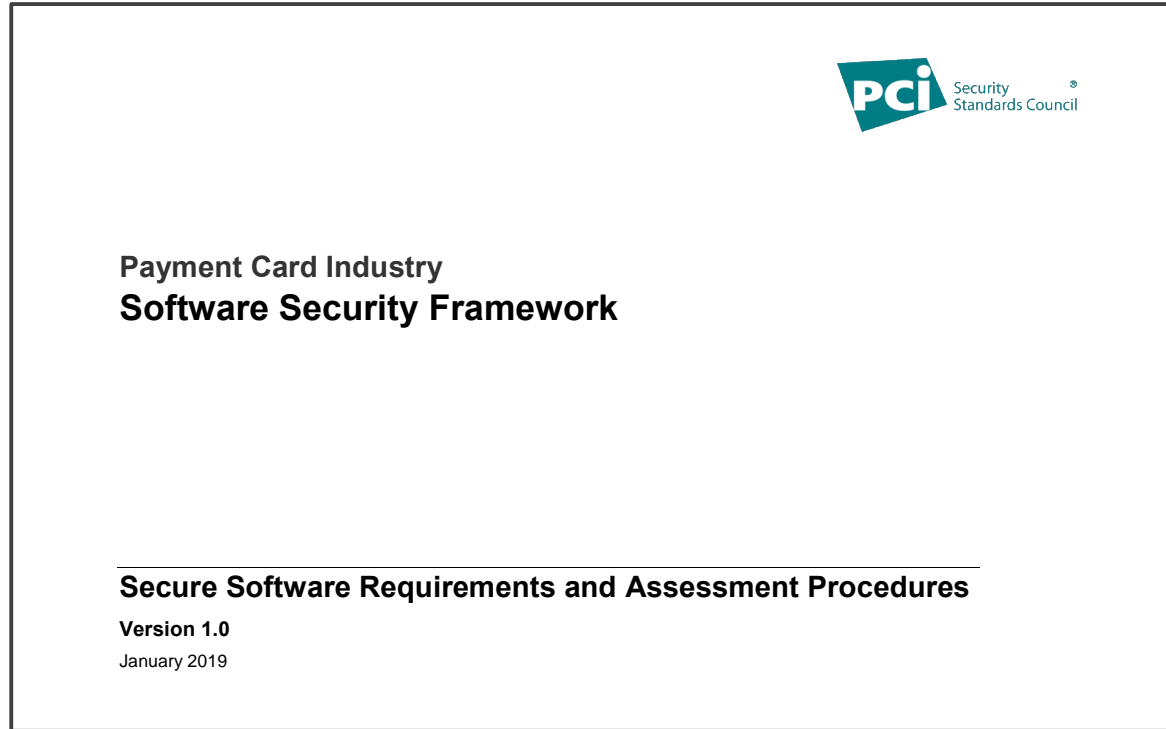
Least privilege access (7)

Authentication and authorization (8)

Logging (10)



PCI Secure Software Standard



www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf



The Software Development Lifecycle

Requirements

+ Security

Design

+ Security

Analysis

+ Security

Build

+ Security

Test

+ Security

Release

+ Security



6.3 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software-development processes Secure SDLC
Examine records	!	Proof the SDLC is being used
Interview people	Y	Software developers





Requirement 6.3.1

Remove development credentials before applications are released





Requirement 6.3.1

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



Requirement Guidance

- Development, test and/or custom application accounts, user IDs, and passwords should be removed from production code before the application becomes active or is released to customers, since these items may give away information about the functioning of the application. Possession of such information could facilitate compromise of the application and related cardholder data.



6.3.1 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software development procedures
Examine records	-	
Interview people	Y	Responsible people





Requirement 6.3.2

Review custom code before release





Requirement 6.3.2

Review custom code prior to release to identify any potential coding vulnerability (using **either manual** or **automated** processes). To include the following:

1. Knowledgeable review
2. Tests according to rules
3. Code fixed before release
4. Results reviewed and approved by management



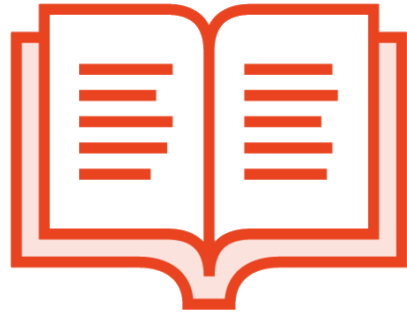
Requirement Guidance

- An individual **knowledgeable** and **experienced** in code-review techniques should be involved in the review process. Code reviews should not be performed by the developer to allow for an independent, objective review. Automated tools or processes may be used, but some coding issues may not be picked up.
- **Correct code errors before deploying** the code to prevent the code exposing the environments to potential exploit.
- **Formal review** helps assure the code was developed in accordance with policies and procedures.

Review Custom Code



1: Independent,
knowledgeable
review



2: Based on the
organization's
standards



3: Errors found
are fixed



4: Management
approval

6.3.2 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software development procedures
Examine records	Y	Recent custom application changes
Interview people	Y	Responsible people





Requirement 6.5

Train developers

Have and use secure coding guidelines





Requirement 6.5

Address **common coding vulnerabilities** in software-development processes as follows:

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



Requirement Guidance

- Requirements 6.5.1 - 6.5.10 are the minimum controls essential; organizations should incorporate the applicable secure coding practices to the technology in their environment.
- Application developers should be trained to identify and resolve issues related to common coding vulnerabilities. Training for developers may be provided in-house or by third parties and should be applicable for technology used.
- **As industry-accepted secure coding practices change, organizational coding practices and developer training should likewise be updated to address new threats.**

6.5 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software development policies and procedures
Examine records	Y	Training records
Interview people	-	



Play by Play: OWASP Top 10 2017

By Troy Hunt and Andrew van der Stock

In this course, you'll learn the risks that made the 2017 OWASP Top 10 and how best to utilize the OWASP Top 10 in your organization.

START A FREE 10-DAY TRIAL

▶ PLAY COURSE OVERVIEW

The OWASP Top 10 2017

🔒 A1: Injection	4m
🔒 A2: Broken Authentication	8m
🔒 A3: Sensitive Data Exposure	4m
🔒 A4: XML External Entities (XXE)	1m
🔒 Employing OWASP ZAP to Exploit XXE	10m
🔒 A5: Broken Access Control	3m
🔒 A6: Security Misconfiguration	1m
🔒 A7: Cross-Site Scripting (XSS)	2m



Requirements 6.5.1 – 6.5.6

Verify processes are in place to protect all applications from these common coding errors



Requirement 6.5.1

Injection flaws, particularly **SQL injection**. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws.



Requirement Guidance

- Injection flaws are a commonly used method for compromising applications. SQL injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data, and allows the attacker to attack components inside the network through the application, to initiate attacks such as buffer overflows, or to reveal both confidential information and server application functionality.
- Information should be validated before being sent to the application.



Requirement 6.5.2

Buffer overflows



Requirement Guidance

- Buffer overflows occur when an application does not have appropriate bounds checking on its buffer space. This can cause the information in the buffer to be pushed out of the buffer's memory space and into executable memory space. When this occurs, the attacker has the ability to insert malicious code at the end of the buffer and then push that malicious code into executable memory space by overflowing the buffer. The malicious code is then executed and often enables the attacker remote access to the application and/or infected system.



Requirement 6.5.3

Insecure cryptographic storage



Requirement Guidance

- Applications that do not utilize strong cryptographic functions properly to store data are at increased risk of being compromised, and exposing authentication credentials and/or cardholder data. If an attacker is able to exploit weak cryptographic processes, they may be able to gain clear-text access to encrypted data.





Requirement 6.5.4

Insecure communications



Requirement Guidance

- Applications that fail to adequately encrypt network traffic using strong cryptography are at increased risk of being compromised and exposing cardholder data. If an attacker is able to exploit weak cryptographic processes, they may be able to gain control of an application or even gain clear-text access to encrypted data.





Requirement 6.5.5

Improper error handling



Requirement Guidance

- Applications can unintentionally leak information about their configuration or internal workings, or expose privileged information through improper error handling methods. Attackers use this weakness to steal sensitive data or compromise the system altogether. If a malicious individual can create errors that the application does not handle properly, they can gain detailed system information, create denial- of-service interruptions, cause security to fail, or crash the server.



Requirement 6.5.6

All “high risk” vulnerabilities identified in the vulnerability identification process (as defined in PCI DSS Requirement 6.1).



Requirement Guidance

- All vulnerabilities identified by an organization’s vulnerability risk-ranking process (defined in Requirement 6.1) to be “high risk” and that could affect the application should be identified and addressed during application development.
- Web applications, both internally and externally (public) facing, have unique security risks based upon their architecture as well as the relative ease and occurrence of compromise.

6.5.1 – 6.5.6 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software-development policies and procedures
Examine records	-	
Interview people	Y	Developers





Requirements 6.5.7 – 6.5.10

Verify processes are in place to protect all **web** applications from these common coding errors



Requirement 6.5.7

Cross-site scripting (XSS)



Requirement Guidance

- XSS flaws occur whenever an application takes user-supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser, which can hijack user sessions, deface web sites, possibly introduce worms, etc.



Requirement 6.5.8

Improper access control
(such as insecure direct object references, failure to restrict URL access, directory traversal, and failure to restrict user access to functions).



Requirement Guidance

- Consistently enforce access control in presentation layer and business logic for all URLs. Frequently, the only way an application protects sensitive functionality is by preventing the display of links or URLs to unauthorized users.
- If user interfaces permit access to unauthorized functions, this access could result in unauthorized individuals gaining access to privileged credentials or cardholder data. Limiting access to data resources will help prevent cardholder data from being presented to unauthorized resources.



Requirement 6.5.9

Cross-site request forgery (CSRF)



Requirement Guidance

- A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then enables the attacker to perform any state-changing operations the victim is authorized to perform (such as updating account details, making purchases, or even authenticating to the application).



Requirement 6.5.10

Broken authentication and session management.



Requirement Guidance

- Secure authentication and session management prevents unauthorized individuals from compromising legitimate account credentials, keys, or session tokens that would otherwise enable the intruder to assume the identity of an authorized user.

6.5.7 – 6.5.10 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Software-development policies and procedures
Examine records	-	
Interview people	Y	Developers





Requirement 6.6

Protect public-facing web apps by either using a Web Application Firewall (WAF) or by an application vulnerability assessment





Requirement 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 reviewing **using application vulnerability security assessment tools** or methods, at least annually and after any changes, or by installing an **automated technical solution**.



Requirement Guidance

- The requirement for reviewing applications or installing web-application firewalls is intended to reduce the number of compromises on public-facing web applications due to poor coding or application management practices.
- Manual or automated vulnerability security assessment tools or methods review and/or test the application for vulnerabilities.
- Web-application firewalls filter and block non-essential traffic at the application layer. Used in conjunction with a network-based firewall, a properly configured web-application firewall prevents application-layer attacks if applications are improperly coded or configured.

6.6 Testing Procedures – Vuln Assessment

Observe/examine systems and settings	-	
Examine documentation	Y	Processes
Examine records	Y	Application security assessments
Interview people	Y	Responsible people



6.6 Testing Procedures – Technical Solution

Observe/examine systems and settings	Y	System configuration settings
Examine documentation	-	
Examine records	-	
Interview people	Y	Responsible people





Requirement 6.7

Have policies and procedures





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 Guidance

- Personnel need to be aware of and following security policies and operational procedures to ensure systems and applications are securely developed and protected from vulnerabilities on a continuous basis.

6.7 Testing Procedures

Observe/examine systems and settings	-	
Examine documentation	Y	Security policies, Operational procedures
Examine records	-	
Interview people	Y	People who should know ...



That's Fine in Theory

