Payment Card Industry (PCI)

Software Security Framework

Secure Software Lifecycle
Requirements and Assessment Procedures

Version 1.1

February 2021

Document Changes

Date	Version	Description
January 2019	1.0	Initial release
February 2021	1.1	Update from v1.0 to address errata and to align with changes to the *Secure SLC Program Guide* to support program expansion. See *Software Security Framework – Summary of Changes from Secure Software Lifecycle Requirements and Assessment Procedures Version 1.0 to 1.1* for details of changes.

Table of Contents

Introduction

This document, the *PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures* (hereafter referred to as the “PCI Secure SLC Standard”), provides a baseline of security requirements with corresponding assessment procedures and guidance to help software vendors design, develop, and maintain secure software throughout the software lifecycle.

The PCI Secure SLC Standard is intended for use as part of the PCI Software Security Framework. Under this framework, software vendors wishing to validate their software lifecycle management practices to this *PCI Secure SLC Standard* may optionally choose to do so. For more information on PCI Secure SLC validation, please refer to the *PCI Secure SLC Program Guide*.

Terminology

A list of applicable terms and definitions is provided in the *PCI Software Security Framework Glossary of Terms, Abbreviations, and Acronyms,* available in the PCI SSC Document Library: https://www.pcisecuritystandards.org/document_library.

Additionally, definitions for general PCI terminology are provided in the *PCI Glossary* on the PCI SSC website at: https://www.pcisecuritystandards.org/pci_security/glossary.

PCI Secure Software Lifecycle Requirements

The security requirements defined within the *PCI Secure SLC Standard* (hereafter referred to as “PCI Secure SLC Requirements”) expand on traditional software development lifecycle (SDLC) models by introducing security concepts and activities throughout the entire software lifecycle—including the design, development, deployment, and maintenance phases. The security concepts described within this document are intended to help software vendors protect sensitive data, minimize vulnerabilities, and defend software from attacks throughout the entire software lifecycle.

Scope of Requirements

The PCI Secure SLC Requirements apply to the software vendor's processes, technology, and personnel involved in the design, development, deployment, and maintenance of the software vendor's software products and services including, but not limited to:

- The policies and processes that govern how the software vendor manages its software throughout the software lifecycle.
- The tools, technologies, and techniques used by the software vendor in the development and management of its software.
- The software-testing methods and technologies used by the software vendor and the results of such testing.
- All people involved in the management of the software vendor's software, including applicable vendor personnel and third-party contributors.
- All processes supporting the software vendor's software lifecycle management activities, including change management, vulnerability management, and risk management.
- The software vendor's software versioning methodology.
- All guidance the software vendor is expected to provide its customers and other stakeholders to ensure that customers know how to implement and configure its software in a secure manner.
- All software vendor communications to its stakeholders.

Some software vendors may have multiple software products covered by different software lifecycle management programs. Prior to assessment against the PCI Secure SLC Requirements, software vendors should identify the software products and associated software lifecycle management program(s) to be covered under the assessment. For more information on defining the scope of the Secure SLC assessment, refer to the *PCI Secure SLC Program Guide*.

Objective-Based Approach to Requirements

There are numerous mature, secure software lifecycle management methodologies and frameworks available that, when properly implemented and maintained, can produce secure software[1]. In recognition of this, the PCI Software Security Framework has adopted an "objective-based" approach to defining the PCI Secure SLC Requirements. This approach acknowledges that there is no "one size fits all" method to software security, and that software vendors need flexibility to determine the secure software lifecycle management practices and methods most appropriate to address their specific business and software risks.

For this approach to be successful, software vendors must possess a robust risk-management practice as an integral part of their "business as usual" operational processes. The specific security controls needed to meet certain requirements in this standard—for example, additional data elements identified by the software vendor as sensitive data[2]—will depend on the software vendor's risk-management priorities and processes. While this approach provides the software vendor with flexibility to implement the appropriate security controls based on identified risk, the software vendor must be able to demonstrate how the implemented controls are supported by the results of its risk-management practices. Without a robust risk-management practice and evidence to support risk-based decision making, adherence to the PCI Secure SLC Requirements may be difficult to validate.

Where a PCI Secure SLC Requirement does not define a specific level of rigor or a minimum frequency for periodic or recurring activities—for example, the required frequency the software vendor must review security strategy performance—the software vendor may define the level of rigor or frequency as appropriate for its business. The rigor and frequency defined by the software vendor must be supported by documented risk assessments and the resultant risk-management decisions. The software vendor must be able to demonstrate that its implementation provides ongoing assurance that the activities are effective and meet the intent of all applicable security requirements.

Equally important is the need for software vendors to understand all the security requirements in this document and consider how the software vendor's software security controls and processes work together as a whole to satisfy the security requirements rather than focusing on any single requirement in isolation.

[1] Examples of other secure software lifecycle management methodologies and frameworks include works from NIST, ISO, SAFECode, Synopsis (BSIMM), and OWASP (OpenSAMM). Please refer to these sources for more information about their methodologies.

[2] Refer to the *PCI Software Security Framework Glossary of Terms, Abbreviations, and Acronyms* for definition of sensitive data.

Requirements Overview

The PCI Secure SLC Requirements are organized into four main sections:

1. Software Security Governance
2. Secure Software Engineering
3. Secure Software and Data Management
4. Security Communications

Within each of the sections defined above, the PCI Secure SLC Requirements are further subdivided into the following components:

- **Control Objectives** – The security outcomes that must be achieved. While all control objectives must be met in order to be validated to this *PCI Secure SLC Standard*, software vendors may define the specific controls, tools, methods and techniques they use to meet each control objective.
- **Test Requirements** –The validation activities to be performed by an assessor to confirm whether a specific control objective has been met. If an assessor determines that alternative testing methods are appropriate to validate a particular control objective, they must justify and document their testing approach as described in the Assessment Procedures and Test Requirements section.
- **Guidance** – Additional information to help software vendors and assessors further understand the intent of each control objective and how it could be met. The guidance may include best practices to be considered as well as examples of controls or methods that, when properly implemented, could meet the intent of the control objective. This guidance is not intended to preclude other methods that a software vendor may use to meet a control objective, nor does it replace or amend the control objective to which it refers.

Assessment Procedures and Test Requirements

To facilitate validation of the software vendor's software lifecycle management practices, software vendors must produce appropriate evidence that confirms they have satisfied the control objectives defined within this standard. The test requirements identified for each control objective describe the expected activities to be performed to validate whether the software vendor has met the objective. Where sub-bullets are specified in a control objective or test requirement, each bullet must be satisfied as part of the validation. In addition, where terms such as "periodic," "appropriate," and "reasonable" are used in the test requirement, it is the software vendor's responsibility to define and defend its decisions regarding the frequency, robustness, and maturity of the implemented controls or processes.

Test requirements typically include the following activities:

- **Examine:** The assessor critically evaluates data evidence. Common examples of evidence include software design and architecture documents (electronic or physical), source code, configuration and metadata files, bug tracking data and other output from software development systems, and security-testing results.
- **Observe**: The assessor watches an action or views something in the environment. Examples of observation subjects include personnel performing tasks or processes, software or system components performing a function or responding to input, system configurations/settings, environmental conditions, and physical controls.
- **Interview**: The assessor converses with individual personnel. The purpose of such interviews may include determining how an activity is performed, whether an activity is performed as defined, and whether personnel have particular knowledge or understanding of applicable policies, processes, responsibilities, or concepts.

The test requirements provide both software vendors and assessors with a common understanding of the expected validation activities to be performed. The specific items or processes to be examined or observed and the personnel to be interviewed should be appropriate for the control objective being validated as well as for each software vendor's unique organizational structure, culture, and business practices.

When documenting the assessment results, the assessor identifies the testing activities performed and the result of each activity. While it is expected that an assessor will perform all the test requirements identified for each control objective, it may also be possible for a control objective to be validated using different or additional testing methods. In such cases, the assessor should document and justify why other testing methods were used and how those methods provide at least the same level of assurance as would have been achieved using the test requirements defined in this standard.

Sampling

Where appropriate, the assessor may utilize sampling as part of the testing process. Samples must be a representative selection of the people, processes, and technologies covered by the PCI Secure SLC assessment. The sample size must be sufficiently large to provide the assessor with assurance that the sample accurately reflects the larger population and that controls are implemented as expected.

In all instances where the assessor's finding is based on a representative sample rather than the complete set of applicable items, the assessor should explicitly note this fact, detail the items chosen as samples for the testing, and provide a justification of the sampling methodology used.

Third-Party Service Providers

Software vendors often rely on outsourced, third-party service providers for certain software lifecycle management functions—e.g., for software development (excluding the use of open-source code), performing code reviews or other testing of the software vendor's software, hosting for the software vendor's software development or delivery platforms, or integrating and installing the software vendor's products.

Where a third-party service can affect the software vendor's software lifecycle management practices or the security of the software vendor's software, the applicable PCI Secure SLC requirements will need to be identified and implemented for that service. The software vendor and service provider will need to understand which software lifecycle management functions are affected by the service provider and identify which PCI Secure SLC Requirements are the responsibility of the service provider and which are the responsibility of the software vendor.

The software vendor is expected to have processes in place to manage risks associated with third-party service providers, including (as applicable for each service):

- Performing due diligence prior to engagement;
- Clear definition of security responsibilities;
- Periodic verification that agreed-upon responsibilities are being met; and
- A written agreement to ensure both parties understand and acknowledge their security responsibilities.

While the ultimate responsibility for the security of the software lies with the software vendor, service providers may be required to demonstrate compliance with the applicable PCI Secure SLC Requirements based on the provided service. The service provider may do so by either:

(a) Undergoing its own PCI Secure SLC assessment for the applicable product(s) or service(s) provided to the software vendor, and providing evidence to the software vendor that demonstrates its compliance to the applicable Secure SLC requirements for that product/service; or

(b) Having the applicable product(s) or service(s) included in the software vendor's PCI Secure SLC assessment, and allowing the software vendor's assessor to evaluate whether the product/service meets the applicable PCI Secure SLC Requirements.

The evidence provided by service providers should be sufficient to verify that the scope of the service provider's PCI Secure SLC assessment covers the services applicable to the software vendor's software lifecycle management practices, and that the relevant PCI Secure SLC Requirements were validated. The specific type of evidence provided will depend on how the assessments are managed. For example, if the service provider undergoes its own PCI Secure SLC assessment, the resulting Report on Compliance (ROC) could provide some or all of the information needed by the software vendor's assessor to validate applicable PCI Secure SLC Requirements. If the service is being included in the software vendor's PCI Secure SLC assessment, the evidence provided would be determined by the control objectives being assessed and the test requirements for those control objectives.

Secure SLC Requirements

Software Security Governance

A formal software security governance program is established to reflect the software vendor's commitment to building secure software and protecting any sensitive data and resources stored, processed, or transmitted by that software.

Control Objectives	Test Requirements	Guidance
Control Objective 1: Security Responsibility and Resources The software vendor's senior leadership team establishes formal responsibility and authority for the security of the software vendor's products and services. The software vendor allocates resources to execute the strategy and ensure that personnel are appropriately skilled.		
1.1 Overall responsibility for the security of the software vendor's products and services is assigned by the vendor's senior leadership team.	**1.1** The assessor shall examine vendor evidence and interview the individual or individuals assigned overall responsibility for the security of the vendor's products and services to confirm the following: • Accountability for ensuring the security of the software vendor's products and services is formally assigned to an individual or team by the software vendor's senior leadership. • Responsibilities include keeping senior leadership informed of security updates, issues, and other matters related to the security of the software vendor's products and services. • Updates are provided to senior leadership at least annually on the performance of and changes to the software vendor's software security policy and strategy described in Control Objective 2.	The formal assignment of responsibility by the software vendor's senior leadership team ensures strategic-level visibility into and influence over the vendor's software security practices. Senior leadership typically represents those individuals or teams with the responsibility and authority to make strategic business decisions for the software vendor organization. In many cases, senior leadership teams are comprised of members of the executive team such as the chief executive officer (CEO), chief financial officer (CFO), chief technology officer (CTO), chief information officer (CIO), chief risk officer (CRO), or similar roles, but that is not the case in all organizations. The distinct structure of the senior leadership team is ultimately determined by the software vendor. Assignment of overall responsibility for the vendor's software security program should include the authority to enforce and execute the organization's software security strategy. Without appropriate authority, those responsible for the security of the software vendor's products and services cannot be reasonably held accountable for ensuring the organization's security strategy is followed. Those responsible for the vendor's software security should provide periodic updates on the state of the vendor's software security program and the performance of its strategy to senior leadership. This allows senior leadership to ensure the strategy is being properly prioritized and resourced, and that changes required as a result of its performance are approved in a timely manner. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
		Evidence to support this control objective might include job descriptions, organization charts, presentations, audio recordings, senior leadership meeting minutes, reports, e-mails, formal communications from senior leadership to the rest of the organization, or any other records that clearly reflect the formal assignment of responsibility and authority, and communications between senior leadership and those responsible for the vendor's software security program regarding program performance.
1.2 Software security responsibilities are assigned.	**1.2.a** The assessor shall examine vendor evidence to confirm the following: • Software security responsibilities are clearly defined and assigned to appropriate individuals or teams, including software development personnel. • Assignment of responsibilities for ensuring the security of the software vendor's products and services covers the entire software lifecycle.	Individuals (including third-party personnel) involved in the design, development, testing and maintenance of the software vendor's products and services should be assigned responsibility and accountability for ensuring that software is designed and maintained in accordance with the software vendor's security strategy and all applicable security requirements, including software-specific requirements. Responsibilities can be assigned to an individual, group, or role; however, individuals assigned to a particular group or role should clearly understand how those software security responsibilities affect their individual job functions, the organization's security expectations, and the individual's role in fulfilling those expectations. Individuals assigned software security responsibilities should be able to demonstrate an understanding of their responsibilities and accountability. Evidence to support this objective might include job descriptions, employee agreements, presentations, company communications, training materials, e-mails, intranet content, or any other documentation or records that clearly and consistently illustrate the assignment of security responsibilities, and the acknowledgement and understanding of those roles and responsibilities.
	1.2.b The assessor shall interview a sample of responsible individuals, including software development personnel, to confirm they are clearly aware of and understand their software security responsibilities.	

Control Objectives	Test Requirements	Guidance
1.3 Software development personnel maintain skills in software security matters relevant to their specific role, responsibility, and job function.	**1.3.a** The assessor shall examine vendor evidence to confirm the following: • A mature process is implemented and maintained for managing and maintaining software security skills for software development personnel. • The skills required for each defined role, responsibility, and job function are clearly defined. • The criteria for maintaining individual skills are clearly defined. • The process includes a review at least annually to ensure software development personnel are maintaining the necessary skills for the security responsibilities they have been assigned.	To be effective in meeting their software security responsibilities, software development personnel must be trained or have experience in performing such responsibilities and must maintain the appropriate skills to properly carry out those responsibilities. At a minimum, all software development personnel must have a basic understanding of general software security concepts and best practices. Individuals with specialized roles and responsibilities should additionally possess specialized skills relevant to the functions they perform. Examples of specialized skills include secure software design (software architects), secure coding techniques (software developers), and security-testing techniques (software testers). Efforts to maintain those skills may include software vendor-provided training, ongoing participation in local or regional user groups, or the achievement and maintenance of industry-specific certifications. It is up to the software vendor to define the necessary criteria for maintaining appropriate job-specific skills and confirm individual adherence at least annually. Evidence to support this control objective might include policies and processes, training materials or content, records of on-the-job training or course attendance, individual qualification certificates, continuing education credits, or any other documentation or evidence that clearly and consistently demonstrates that software development personnel possess and maintain appropriate skills and knowledge for their specific job function and responsibilities.
	1.3.b For a sample of software development personnel, examine vendor evidence and interview personnel to confirm the following: • Individuals have demonstrated that they possess the skills required for their role, responsibility, or job function. • Individuals have satisfied the criteria for maintaining their individual skills.	

Control Objectives	Test Requirements	Guidance
Control Objective 2: Software Security Policy and Strategy The software vendor defines, maintains, and communicates a software security policy and a strategy for ensuring the secure design, development, and management of its products and services. Performance against the software security strategy is monitored and tracked.		
2.1 Regulatory and industry security and compliance requirements applicable to the software vendor's operations, products, and services and the data stored, processed, or transmitted by the software vendor are identified and monitored.	**2.1** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists to identify and monitor external regulatory and industry security and compliance requirements. • The process includes reviewing sources of regulatory and industry security and compliance requirements for changes at least annually. • The process results in an inventory of external regulatory and industry security and compliance requirements. • The inventory is updated as external security and compliance requirements change.	Many organizations are subject to requirements for the protection of certain types of information and data such as personally identifiable information (PII), cardholder data (CHD), and protected health information (PHI). Software vendors should maintain awareness of evolving industry and regulatory requirements applicable to their operations and products. Maintaining ongoing awareness of external security and compliance obligations allows the software vendor to ensure its processes adequately address those requirements at all times, including whenever those requirements are updated or new requirements are introduced. Evidence to support this control objective might include documented policies and processes, internal standards, requirement mappings, internal presentations, training materials, or any other documentation or records that clearly and consistently illustrate that the software vendor has made reasonable efforts to understand and monitor its external security and compliance requirements.
2.2 A software security policy is defined and establishes the specific rules and goals for ensuring the software vendor's products and services are designed, developed, and maintained to be secure, resistant to attack, and in a manner that satisfies the software vendor's security and compliance obligations.	**2.2.a** The assessor shall examine vendor evidence to confirm the following: • A software security policy exists and is communicated to appropriate software vendor personnel and business partners, including all software development personnel. • At a minimum, the policy covers all control objectives within this standard (either explicitly or implicitly). • The policy is defined in sufficient detail such that the security rules and goals are measurable. • The software vendor's senior leadership team has approved the software security policy.	Software vendors must establish a company-wide software security policy to ensure that all individuals or teams—including relevant business partners—involved in software design, development, and maintenance are aware and have a consistent understanding of how the vendor's software products and services should be securely built and maintained, and how any critical assets should be handled. The software security policy (or policies) should be known and thoroughly understood by those with the responsibility to ensure they are met, as well as those individuals and teams who have the ability to affect the security of the software vendor's products and services. The software vendor's senior leadership team should openly support the establishment and enforcement of the software security policy through appropriate communications to software vendor personnel, to reinforce the importance of software security to the vendor organization and its leadership. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	2.2.b The assessor shall interview a sample of software development personnel to confirm they are aware of and understand the software security policy.	Evidence to support this control objective might include documented policies and processes, presentations, mission statements, e-mails, company intranet content, or other formal company communications that clearly and consistently illustrate efforts to ensure appropriate personnel and business partners are aware of and understand the vendor's software security policy.
2.3 A formal software security strategy for ensuring the security of the software vendor's products and services and satisfying its software security policy is established and maintained.	**2.3.a** The assessor shall examine vendor evidence and interview responsible personnel to confirm the following: • A strategy for ensuring the security of the software vendor's products and services is defined. • The software security strategy clearly outlines how the software security policy is to be satisfied. • The software security strategy is based on or aligned with industry-accepted methodologies. • The software security strategy covers the entire lifecycle of the software vendor's products and services. • The software security strategy is communicated to appropriate personnel, including software development personnel. • The software security strategy is reviewed at least annually and updated as needed (such as when business needs, external drivers, and products and services evolve).	A software security strategy is a high-level plan, roadmap, or methodology for ensuring the secure design, development, and maintenance of the software vendor's products and services, and adherence to the software vendor's software security policy. Software vendors should either adopt existing or develop their own frameworks or methodologies in accordance with industry-accepted practices for secure software lifecycle management. By aligning its software security strategy with industry-accepted methodologies, the software vendor is less likely to overlook important aspects of secure software lifecycle management. Software vendors that develop their own methodologies should understand how they differ from industry-accepted methodologies, identify any gaps, and ensure that sufficient evidence is maintained to clearly illustrate how their methodologies are at least as effective as those accepted by the industry. Examples of industry-accepted methodologies that are commonly used as benchmarks for secure software development and management include, but are not limited to, current versions of: • *ISO/IEC 27034 Application Security Guidelines* • *Building Security In Maturity Model (BSIMM)* • *OWASP Software Assurance Maturity Model (OpenSAMM)* • *NIST Special Publication 800-160 and its Appendixes*

Control Objectives	Test Requirements	Guidance
	2.3.b The assessor shall interview a sample of software development personnel to confirm they are aware of and understand the software security strategy.	The software security strategy should evolve as internal factors—such as the software vendor's business strategy or product/service offerings—or external factors—such as external security and compliance requirements—evolve. Therefore, the software security strategy is not static and should be periodically reviewed and updated to maintain alignment with business needs and priorities. Evidence to support this requirement might include documented security plans or methodologies, presentations, policies and processes, training materials, meeting minutes, interviewer notes, e-mails or executive communications, mappings, or references to industry-accepted methodologies, gap analysis results, or any other records or documentation that clearly and consistently illustrates that the vendor has made a reasonable effort to develop, maintain, and keep current a formal strategy for satisfying the vendor's software security policy.
2.4 Software security assurance processes are implemented and maintained throughout the entire software lifecycle. ***Note:*** *This control objective focuses on the overall management of security assurance processes and provides the foundation for specific assurance processes defined within this document.*	**2.4.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • Software security assurance processes are defined, implemented and maintained. • An inventory of software security assurance processes is maintained.	Software security assurance processes are activities that are implemented to carry out the software vendor's software security strategy and to facilitate secure software design, development, and maintenance. To ensure that security and compliance requirements are met, software security policy is satisfied, and the software vendor's products and services are secure and resistant to attack, software vendors need to define such processes throughout all phases of the software lifecycle. These may include security "checkpoints," which are distinct points within the software development process where software is checked to make sure security requirements are met. Examples of software security assurance processes and controls include software-design reviews, automated code reviews, security-specific functional testing, and change-management processes. For organizations that leverage Agile software development methodologies, security checkpoints may be incorporated into the "story" acceptance criteria or the criteria for determining when work is considered "done." *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	2.4.b For a sample of software security assurance processes, the assessor shall examine vendor evidence and interview personnel to confirm the following: • Software security assurance processes clearly address the specific rules and goals within the software vendor's software security policy. • Software security assurance processes are aligned with the software vendor's software security strategy. • Software vendor personnel, including software development personnel, are assigned responsibility and accountability for the execution and performance of the security assurance process in accordance with Control Objective 1.2. • The individuals or teams responsible for performing and maintaining each security assurance process are clearly aware of their responsibilities. • The results or outcomes of each security assurance process are monitored in accordance with Control Objective 2.6.	Evidence to support this requirement might include documented policies and processes, security-control inventories, output from Governance Risk and Compliance (GRC) or other management tools, software-specific requirements documentation, or any other evidence that clearly and consistently identifies the software security assurance processes that have been implemented and illustrates that the security assurance processes are appropriate for the function they are intended to provide. Additionally, evidence to illustrate the software security assurance processes are implemented properly may include system or process outputs such as threat models, security test results, bug tracking data, audit log data, incident response, etc.
2.5 Evidence is generated and maintained to demonstrate the effectiveness of software security assurance processes.	**2.5.a** The assessor shall examine vendor evidence, including the inventory of software security assurance processes described in Test Requirement 2.4.a, and interview personnel to confirm that evidence is generated and maintained for each security assurance process.	To demonstrate the effectiveness of software security assurance processes, evidence should be generated and maintained for each process to illustrate that it directly results in or contributes to the expected security outcomes—e.g., fewer vulnerabilities or greater resistance to attacks. Evidence needs to be frequently collected and kept up to date to ensure it accurately reflects the ongoing effectiveness of security assurance processes. Without a track record of performance for software security assurance processes, it becomes almost impossible to effectively perform root-cause analysis when such processes fail to produce the expected results. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	2.5.b For a sample of security assurance processes, the assessor shall examine evidence and other output from the processes and interview personnel to confirm the evidence generated for each process reasonably demonstrates the process is operating effectively and as intended.	Evidence to support this objective might include security control and evidence generation inventories, vulnerability reports, penetration testing results, or any other records and evidence that clearly and consistently illustrates evidence is generated for each software security assurance process and that the evidence clearly illustrates the effectiveness of the processes.
2.6 Failures or weaknesses in software security assurance processes are detected. Weak or ineffective security assurance processes are updated, augmented or replaced.	**2.6.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists to detect and evaluate weak or ineffective security assurance processes. • The criteria for determining a weak or ineffective security assurance process is defined and justified. • Security assurance processes are updated, augmented or replaced when deemed weak or ineffective.	Software vendors should monitor their security assurance processes to confirm that they remain appropriate (i.e., fit for purpose) and effective for their intended purpose and function. For example, the use of manual code reviews may be sufficient to detect all coding errors and vulnerabilities for software with a very limited code base. However, as the code base grows, the use of manual code reviews for the same purpose becomes increasingly impractical or insufficient, and automated testing tools (such as automated static-code scanners and dynamic software-analysis tools) should be utilized. One method for detecting weak or ineffective security controls is to define a set of metrics or trends that can be used to measure the effectiveness of security assurance processes. For example, the results from a vendor's security testing may provide greater insight into the effectiveness of security assurance processes. If security tests repeatedly find vulnerabilities within the software, it may be an indication that applicable security assurance processes are not being executed properly or working as intended. Another method to detect weak or ineffective security assurance processes would be to perform regular reviews of those processes and the evidence generated by those processes to verify they continue to be appropriate for their intended purpose. Evidence to support this requirement might include process-generated evidence, security test results, root-cause analysis, documented remediation actions, or any other evidence that clearly and consistently illustrates that the effectiveness of software security assurance processes is monitored, failures and weaknesses are detected, and security assurance processes are updated, augmented or replaced when no longer effective at satisfying their intended purpose.
	2.6.b For a sample of the security assurance processes identified in Control Objective 2.4, the assessor shall interview personnel and examine any additional evidence necessary to determine if any failures or weaknesses in those security processes occurred, and to confirm that weak or ineffective processes were updated, augmented or replaced.	

Secure Software Engineering

Software is designed and developed to protect critical software assets and to be resistant to attacks.

Control Objectives	Test Requirements	Guidance
Control Objective 3: Threat Identification and Mitigation The software vendor continuously identifies, assesses, and manages risk to its software and services.		
3.1 Critical assets are identified and classified.	**3.1** The assessor shall examine vendor evidence to confirm the following: • A mature process exists to identify and classify critical assets. • The criteria for identifying critical assets and determining the confidentiality, integrity, and resiliency requirements for each critical asset are defined. • The process accounts for all types of critical assets—including sensitive data, sensitive resources, and sensitive functions—for the vendor's software. • The process results in an inventory of critical assets used by the vendor's software.	Before the software vendor can determine how to effectively secure and defend its software against attacks, it must first develop a thorough understanding of the software's critical assets that could be targeted by attackers. Critical assets include any sensitive data collected, stored, processed, or transmitted by the vendor's software, as well as any sensitive functions and sensitive resources within or used by the software. Examples of analysis techniques that could be used to identify critical assets include, but are not limited to, Mission Impact Analysis (MIA), Functional Dependency Network Analysis (FDNA), and Mission Threat Analysis.

Control Objectives	Test Requirements	Guidance
3.2 Threats to the software and weaknesses within its design are continuously identified and assessed.	**3.2.a** The assessor shall examine vendor evidence, including process documentation and assessment results to confirm the following: • A mature process exists to identify, assess, and monitor software threats and design weaknesses (i.e., flaws). • The assessment accounts for all software inputs/outputs, process/data flows, trust boundaries and decision points, and how they may be exploited by an attacker. • The assessment accounts for the entire code base, including how the use of third-party, open-source, or shared components or libraries, APIs, services, and applications used for the delivery and operation of the software may be leveraged in an attack. • The assessment results in a recorded inventory of threats and design flaws. • Assessments are routinely performed to account for changes to existing or the emergence of new threats or design flaws.	Determining how to effectively secure and defend software against attacks requires a thorough understanding of the specific threats and vulnerabilities applicable to the vendor's software. This typically involves understanding: • The motivations an attacker may have for attacking software; • Weaknesses in the software design that an attacker might attempt to exploit; • The exploitability of identified weaknesses; and • The impact of a successful attack. This information helps the software vendor to identify the threats and design flaws that present the most significant and immediate risk, and to prioritize remediation activities necessary to address them. Information regarding software threats can be obtained from a variety of sources, both external and internal. Examples of external sources include publications from organizations such as SANS, MITRE, and CERT that specialize in tracking common system vulnerabilities and attack techniques, or industry-specific sources that provide threat intelligence for specific sectors, such as FS-ISAC for the financial services industry and R-CISC for the retail industry. Other external sources of threat information and design weaknesses could include technology vendors, open-source user communities, industry publications, and academic papers. Internal sources could include reports from internal research and design teams, formal threat models, or actual activity data from internal security or operations teams. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	3.2.b Where open-source software components are utilized as part of the software, the assessor shall examine vendor evidence, including process documentation and assessment results to confirm these components are managed as follows: • An inventory of open-source components used in the vendor's software is maintained. • A mature process exists to analyze and mitigate the use of open-source components with known vulnerabilities. • The software vendor monitors vulnerabilities in open-source components throughout their use or inclusion in the vendor's software. • An appropriate patching strategy for open-source components is defined.	Where open-source software components are used, the software vendor should consider any risks associated with the use of the open-source components and the extent to which the open-source software provider manages the security of those components. Additionally, the software vendor will need to confirm that support—including up-to-date security patches—is available (whether provided by an internal or external entity) for the open-source component. The use of open-source components should be supported by a clear policy about how those components are evaluated and implemented. A reliable support system should be in place to identify errors or problems and evaluate and address them in a timely manner. Where vulnerabilities are identified in open-source components that are applicable to their software, software vendors should have processes in place to analyze those vulnerabilities and update the components to appropriate, non-vulnerable versions in a timely manner. When patches for open-source components are no longer available, those components should be replaced by actively supported ones. Vendors should identify and establish sources and processes for managing vulnerabilities in open-source components that are appropriate for their software design and release frequency.
	3.2.c For a sample of vendor software, the assessor shall examine assessment results for the selected software to confirm the following: • All software inputs/outputs, process/data flows, trust boundaries and decision points were considered during the assessment. • The entire code base, including how the use of third-party, open-source or shared components or libraries, APIs, services, and applications used for the delivery and operation of the software were considered during the assessment.	

Control Objectives	Test Requirements	Guidance
3.3 Software security controls are implemented in the software to mitigate threats and design weaknesses.	**3.3.a** The assessor shall examine vendor evidence, including process documentation and software-specific threat and design information, to confirm the following: • A mature process exists for defining software-specific security requirements and implementing software security controls within the software to mitigate software threats and design flaws. • Decisions on whether and how to mitigate a specific threat or design flaw are recorded, justified, and approved by appropriate personnel. • Any remaining residual risk is recorded, justified, and approved by appropriate personnel.	To ensure that its software is resistant to attacks, software vendors must implement software-specific controls or countermeasures in their software to mitigate the specific threats and design weaknesses. Examples of such controls include the use of multi-factor authentication mechanisms to prevent unauthorized individuals gaining access to critical assets, and logging mechanisms to detect if and when authentication mechanisms might have been circumvented. Other examples include the use of input validation routines or parameterized queries to protect software from SQL-injection attacks. Except where specific software security controls and countermeasures are defined within this standard, it is up to the vendor to determine the most appropriate software security controls to implement. The specific controls used will be dependent on the software threats identified in Control Objective 3.2 as well as the software's architecture, the software's intended function, the data it handles, and the external resources it utilizes. Evidence to support this control objective may include software-specific requirements documentation, feature lists, security control inventories, change-management documentation, risk assessment reports, test results, or any other evidence or information that clearly and consistently illustrates that the software vendor implements and maintains security controls in software to address the risks to that software.
	3.3.b The assessor shall examine evidence and interview personnel to confirm the following: • Decisions on whether and how to mitigate a specific threat or design flaw are reasonably justified. • Any remaining residual risk is reasonably justified.	
	3.3.c The assessor shall examine vendor evidence to confirm that security controls have been implemented to mitigate all identified threats and design flaws.	

Control Objectives	Test Requirements	Guidance
3.4 Failures or weaknesses in software security controls are detected. Weak or ineffective security controls are updated, augmented or replaced.	**3.4.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists to identify weak or ineffective software security controls and to update, augment, or replace them. • The criteria for determining a weak or ineffective security control is defined and justified. • The process involves monitoring security control effectiveness throughout the software lifecycle. • Weak or ineffective security controls are updated, augmented, or replaced in a timely manner upon detection.	Vendors should monitor and/or routinely test their software to confirm that implemented software security controls remain appropriate (i.e., fit for purpose) and effective for sufficiently mitigating evolving risks or design flaws. For example, a software-specific security requirement may call for cryptography to be used to protect software communications. While the use of SSL may have been sufficient upon the initial design and release of the software, SSL is no longer sufficient to adequately protect communications as new threats and attack methods have significantly reduced its effectiveness as a security control. Therefore, it is imperative that vendors have processes in place to continuously monitor implemented security controls to make sure that they remain appropriate and sufficient to mitigate evolving threats and design flaws throughout the entire lifetime of the software. Evidence to support this requirement might include software-specific documentation, features lists, software-specific security control inventories, change-management documentation, risk-assessment reports, penetration test results, output from active monitoring systems, bug bounty program data, or any other evidence or information that clearly and consistently illustrates that the effectiveness of software security controls is monitored and that software-specific software security controls are updated, augmented, or replaced when no longer effective at satisfying their intended purpose of resisting attacks.
	3.4.b The assessor shall examine vendor evidence, including software-specific data or test results, and details of software-specific updates to confirm the following: • Security controls that have been deemed “weak” or “ineffective” have been updated, augmented or replaced. • Decisions on whether and how to replace and augment weak or ineffective security controls are made in accordance with defined criteria and with Control Objective 3.3.	

Control Objectives	Test Requirements	Guidance
Control Objective 4: Vulnerability Detection and Mitigation The software vendor detects and mitigates vulnerabilities in software to ensure that its software remains resistant to attacks throughout its entire lifecycle.		
4.1 Existing and emerging software vulnerabilities are detected in a timely manner.	**4.1.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists for testing software for the existence and emergence of vulnerabilities (i.e., security testing). • Tools or methods used for security testing are appropriate for detecting applicable vulnerabilities in the vendor's software, and are suitable for the software architectures, and the software development languages and frameworks employed. • Security testing is performed throughout the entire software lifecycle, including after release. • Security testing accounts for the entire code base, including detecting vulnerabilities in any third-party, open-source, and shared components and libraries. • Security testing is performed by authorized and objective vendor personnel or third parties. • Security testing results in an inventory of identified vulnerabilities. • Security-testing details including the tools used, their configurations, and the specific tests performed are recorded and retained.	Software should be monitored or routinely tested to confirm that vulnerabilities are identified and mitigated before software or code updates are released into production, and to address any vulnerabilities that may have been discovered since release. Routine security testing should be performed prior to or as part of the code-commit process to detect coding errors or the use of insecure functions. It could also be performed during unit, integration, regression, or interoperability testing, or during separate security testing. Security testing should be performed consistently and throughout all stages of the software lifecycle, including during various pre-release phases of the software development process and after code release, to ensure the software is free from vulnerabilities upon launch and any subsequent updates, and remains free from vulnerabilities throughout its lifetime. Security testing should be performed by appropriately skilled vendor personnel or third parties. In addition, security testing personnel should be able to conduct tests in an objective manner and be authorized to escalate any identified vulnerabilities to appropriate management or development personnel so they can be properly addressed. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	4.1.b The assessor shall examine evidence, including software-specific security testing configurations and test results to confirm the following: • Security-testing tools are configured in a manner that is appropriate for the intended tests performed. • Security testing accounts for the entire code base, including detecting vulnerabilities in any third-party, open-source, and shared components and libraries. • Security testing was performed by authorized and objective vendor personnel or third parties.	Evidence to support this control objective could include software-specific requirements documentation, security test results, feature lists, change-management documentation, entries in the vendor's workflow (bug tracking) database, or any other evidence or information that clearly and consistently shows that security testing is performed routinely to detect vulnerabilities in code prior to release as well as vulnerabilities discovered since code launch.
	4.1.c The assessor shall examine vendor evidence and interview personnel to confirm that personnel responsible for testing are knowledgeable and skilled in the following areas in accordance with Control Objective 1.3: • Software security testing techniques • Security testing tools settings, configurations, and recommended usage	
	4.1.d For a sample of vendor software, examine software-specific testing results to confirm that security testing is performed throughout the software lifecycle.	

Control Objectives	Test Requirements	Guidance
4.2 Newly discovered vulnerabilities are fixed in a timely manner. The reintroduction of similar or previously resolved vulnerabilities is prevented.	**4.2.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists for distributing and deploying fixes for newly discovered vulnerabilities and preventing the reintroduction of previously resolved vulnerabilities. • The process includes methods to prevent previously resolved vulnerabilities or other similar vulnerabilities from being reintroduced into the software. • The criteria for determining the “criticality” or “severity” of vulnerabilities and how to address vulnerabilities are defined and justified. • Fixes to address vulnerabilities in production code are made available and deployed in accordance with defined criteria. • Decisions not to provide fixes in accordance with defined criteria are approved and justified by appropriate personnel on a case-by-case basis.	Vulnerabilities should be addressed in a manner commensurate with the risk they pose to the software or its stakeholders. The most critical or severe vulnerabilities—i.e., those with the highest exploitability and/or the greatest impact to stakeholders—should be patched immediately, followed by those with moderate-to-low exploitability and/or impact. Additionally, the discovery of new classes of vulnerabilities should be used as a source of input for process improvement. Software should be reviewed for instances of similar vulnerabilities, and the vendor’s development processes updated to enable detection and mitigation of such vulnerabilities in the future. In some cases, it may be impractical for a vendor to fix all identified vulnerabilities prior to the release of production code or updates. In such circumstances, the vendor should have a methodology with clear criteria defined for prioritizing vulnerability fixes. The default outcome should always be that vulnerabilities are fixed before the software is released. In cases where it is not possible to fix a vulnerability prior to release, an exception process involving management at a level commensurate with the severity of the vulnerability should be invoked. The process should include documented justification for why a fix for was not provided to address the vulnerability. If it is not possible to mitigate a certain vulnerability prior to release, the vendor should provide stakeholders with additional guidance to mitigate the risk of exploitation until a security update to fix the vulnerability can be made available. Under no circumstances should a previously resolved vulnerability be reintroduced into production code, nor should similar vulnerabilities within the same class of vulnerabilities be reintroduced. Additional assurance processes and safeguards should be implemented to ensure that such incidents are avoided. The specific processes to prevent such occurrences will largely depend on how the vendor’s software is structured and how the vendor manages software updates. It is up to the software vendor to determine the most appropriate methods to prevent the reintroduction of vulnerabilities into production code.
	4.2.b For a sample of vendor software, the assessor shall examine software-specific security-testing results and the details of software updates to confirm that security fixes are made available and deployed (where applicable) in accordance with defined criteria.	
	4.2.c For a sample of vendor software, the assessor shall interview personnel to confirm that decisions not to provide security fixes in accordance with defined criteria are justified by appropriate personnel.	

Secure Software and Data Management

The confidentiality and integrity of software and its critical assets are maintained throughout the software lifecycle.

Control Objectives	Test Requirements	Guidance
Control Objective 5: Change Management The software vendor identifies and manages all software changes throughout the software lifecycle.		
5.1 All changes to software are identified, assessed, and approved.	**5.1.a** The assessor shall examine vendor evidence and interview personnel to confirm: • A mature process exists to identify, assess, and approve all changes to software. • The process includes an analysis of the security impact of all changes. • The process results in an inventory of all changes made to software, including a record of the determined security impact. • All change-management decisions are recorded. • All implemented changes are authorized by responsible personnel. • The inventory of changes identifies the individual creator of the code and individual authorizing the change, for each code change. • All decisions to implement changes are justified.	All changes to software should be defined, documented, approved, and tracked so that any vulnerabilities attributed to such changes may be identified and resolved as quickly as possible. The harder it is to trace vulnerabilities back to the changes that introduced them, the longer it takes to resolve those vulnerabilities—thus placing the software at greater risk of attack or compromise. It is imperative to understand the security risk of a change to the software to ensure that it is addressed accordingly. It often involves understanding the types of software functionality the change impacts (e.g., functionality that deals with encryption or authentication processes), the type of information assets that the functionality can access or manipulate, the likelihood of successful vulnerability exploitation, and the impact a successful attack may have on stakeholders.
	5.1.b For a sample of changes, the assessor shall examine software-specific and change-specific documentation or evidence to confirm the following: • All changes are authorized by responsible personnel. • All decisions to implement the changes are recorded and include justification for the change. • The inventory of changes clearly identifies the individual creator of the code and the individual authorizing the change, for each code change.	

Control Objectives	Test Requirements	Guidance
5.2 All software versions are uniquely identified and tracked throughout the software lifecycle.	**5.2.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A formal system or methodology for uniquely identifying each version of software is defined. • The system or methodology includes arranging unique identifiers or version elements in a sequential and logical manner. • All changes to software functionality are clearly associated with a unique software version.	Without a thoroughly defined versioning methodology, changes to software may not be properly identified, and customers and integrators/resellers may not understand the impact of such changes. The system or methodology adopted by the vendor should allow different release versions of a software product to be easily distinguishable. To ensure a software version accurately represents the release version, the versioning system or methodology should be integrated with applicable lifecycle functions, such as code control and change management. The versioning system or methodology should encompass all changes to all relevant software components. As several iterations of a software component may be produced for a single software release, the versioning system or methodology should easily identify the version of each component associated with a specific software release version. The method used for identifying the software release versions—for example, a version numbering scheme—should be documented and reflect the type of change and its impact on the software. For software intended to be validated under the PCI Software Security Framework, the vendor's versioning system or methodology is important in determining updates to the PCI SSC List of Validated Payment Software. Refer to the *PCI Secure Software Program Guide* for further information.
	5.2.b For a sample of software updates, the assessor shall examine vendor evidence, including change-specific documentation, to confirm the following: • Software versions are updated in accordance with the defined versioning system or methodology. • All changes to software functionality are clearly associated with a unique software version.	

Control Objectives	Test Requirements	Guidance
Control Objective 6: Software Integrity Protection The integrity of software is protected throughout the software lifecycle.		
6.1 The integrity of all software code, including third-party components, is maintained throughout the entire software lifecycle.	**6.1** The assessor shall examine evidence, interview personnel, and observe tools and processes to confirm: • A mature process, mechanism, and/or tool(s) exist to protect the integrity of the software code, including third-party components. • The processes, mechanisms, and/or tools are reasonable and appropriate for protecting the integrity of software code. • Processes, mechanisms, or the use of tools results in the timely detection of any unauthorized attempts to tamper with or access software code. • Unauthorized attempts to tamper with or access software code are investigated in a timely manner.	Effective software-code control practices help ensure that all changes to code are authorized and performed only by those with a legitimate reason to change the code. Examples of these practices include code check-in and check-out procedures with strict access controls, and a comparison—for example, using a checksum—immediately before updating code to confirm that the last approved version has not been changed. It is important that controls cover all software code, third-party components and libraries, configuration files, etc. that are controlled by the vendor. The integrity and confidentiality of these assets need to be maintained, as they often contain sensitive data such as intellectual property—for example, business logic—logic of security functions, configuration of cryptographic functions (e.g., white-box cryptography), etc.
6.2 Software releases and updates are delivered in a secure manner that ensures the integrity of the updated code.	**6.2** The assessor shall examine vendor evidence, interview personnel, and observe tools and processes to confirm the following: • A mature process, mechanism, and/or tool(s) exist to ensure the integrity of software updates during delivery. • The processes, mechanisms, and/or tools are reasonable and appropriate for protecting the update code. • Processes, mechanisms, and/or the use of tools results in the secure delivery of updated code.	Security updates should include a mechanism within the update process to verify the updated code has not been replaced or tampered with. Examples of integrity checks include checksums and properly implemented digitally signed certificates. To ensure the implemented controls are adequate to address the evolving attack vectors, software vendors should perform periodic reviews to confirm their continued effectiveness.

Control Objectives	Test Requirements	Guidance
Control Objective 7: Sensitive Data Protection The confidentiality of sensitive production data is maintained on vendor systems.		
7.1 Sensitive production data is only collected and retained on software vendor systems where there is a legitimate business or technical need.	**7.1.** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists to record and authorize the collection and retention of any sensitive production data. • An inventory of sensitive production data captured or stored by the software vendor's products and services is maintained. • Decisions to use sensitive production data are approved by appropriate software vendor personnel. • Decisions to use sensitive production data are recorded and reasonably justified.	To protect the confidentiality of any sensitive production data—that is, sensitive data that is owned by an entity other than the software vendor— and stored on software vendor systems, such data should never be used for purposes other than those for which the data was originally collected. If the software vendor provides services to its stakeholders that could result in the collection of sensitive production data—for example, for troubleshooting or debugging purposes—then the software vendor should record which specific data elements it collects and retains, and clearly communicate what data elements are collected and why they are collected to its customers and other relevant stakeholders. The inventory of sensitive production data retained by the software vendor should include identification of the specific data elements captured, whether storage of each element is permitted, and the security controls required—for example, to protect confidentiality and/or integrity—for each data element during storage and transmission.

Control Objectives	Test Requirements	Guidance
7.2 Sensitive production data is protected when retained on software vendor systems and securely deleted when no longer needed.	**7.2.a** The assessor shall examine vendor evidence and interview personnel to confirm that a mature process exists to ensure sensitive production data is protected when retained on software vendor systems and is securely deleted when no longer needed.	When software vendors collect sensitive production data from their stakeholders—for example, for debugging or other customer support purposes—the vendor should coordinate with its stakeholders to identify which data elements require protection. Vendor stakeholders may have their own definition and associated security requirements for sensitive data, and appropriate protection efforts should be agreed upon by both parties. Where the software vendor collects or retains sensitive production data, the software vendor should ensure it is secured—for example, by using robust access control measures and/or strong cryptography with industry-accepted key-management processes. As soon as it is no longer needed for its collected purpose, sensitive production data should be securely deleted such that it is not possible to reconstruct or recover the data from any software vendor system.
	7.2.b The assessor shall examine vendor evidence and observe a sample of vendor systems to confirm the following: • Sensitive production data is not resident on software vendor systems unless appropriate evidence of approval and justification exists. • Sensitive production is appropriately protected where it is retained. • Secure deletion processes or mechanisms are sufficient to render sensitive production data irretrievable.	

Security Communications

The software vendor provides timely information to its stakeholders (e.g., customers, installers, integrators, etc.) regarding security issues affecting its software, and thorough guidance on secure software implementation, configuration, operation, and updates.

Control Objectives	Test Requirements	Guidance
Control Objective 8: Software Vendor Implementation Guidance The vendor provides stakeholders with clear and thorough guidance on the secure implementation, configuration, and operation of its software.		
8.1 The software vendor provides stakeholders with clear and thorough guidance on the secure implementation, configuration and operation of its software.	**8.1.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • A mature process exists to produce, maintain, and make available to stakeholders guidance on the secure implementation, configuration, and operation of its software. • The implementation guidance includes documentation of all configurable security-related options and parameters for the vendor’s software, and instructions for properly configuring and securing each of those options and parameters.	When followed, the software vendor's implementation guidance provides assurance that the software and patches are securely installed, configured, and maintained in the customer environment, and that all desired security functionality is active and working as intended. The guidance should cover all options and functionality available to software users that could affect the security of the software or the data it interacts with. The guidance should also include secure configuration options for any components provided with or supported by the software, such as external software and underlying platforms. Examples of configurable options include: • Changing default credentials and passwords • Enabling and disabling application accounts, services, and features • Changes in resource access permissions • Integration with third-party cryptographic libraries, random number generators, etc. Following the secure implementation guidance should result in a secure configuration across all configurable options. *(continued on next page)*
	8.1.b For a sample of vendor software, examine software-specific documentation and materials to confirm that the software vendor provides and maintains guidance on the secure configuration of each security-related option or parameter available in the vendor’s software.	

Control Objectives	Test Requirements	Guidance
8.2 Secure implementation guidance includes detailed instructions on how to securely install, configure, and maintain all software components and supported platforms.	**8.2** The assessor shall examine vendor evidence to confirm the following: • The secure implementation guidance includes instructions on how to securely install or initialize, configure, and maintain the software. • The secure implementation guidance is sufficiently detailed. • Evidence exists or is obtained to illustrate that following the secure implementation guidance results in a secure software configuration.	As the software vendor is expected to continuously identify, assess, and manage risks to its software, the vendor's software-change processes should include determining the impact of the change to software vendor guidance. Software changes that impact a configurable feature or option should result in an update to the secure implementation guidance.
8.3 Secure implementation guidance is aligned with software updates.	**8.3.a** The assessor shall examine vendor evidence and interview personnel to confirm the following: • The process to produce and maintain secure implementation guidance includes generation of updated guidance when new software updates are released, or security-related options or parameters are introduced or modified. • Secure implementation guidance is reviewed at least annually for accuracy even if updates to security-related options and parameters are not issued.	
	8.3.b For a sample of software updates, examine secure implementation guidance as well as details of the software updates to confirm that as security-related options and parameters are updated or added, the secure implementation guidance is updated.	

Control Objectives	Test Requirements	Guidance
Control Objective 9: Stakeholder Communications The software vendor maintains communication channels with stakeholders regarding potential security issues and mitigation options.		
9.1 Communication channels are defined and made available for customers, installers, integrators, and other relevant parties to report and receive information on security issues and mitigation options.	**9.1** The assessor shall examine evidence and interview personnel to confirm the following: • A mature process exists to support open, bi-directional communications with stakeholders for reporting and receiving security information regarding the software vendor’s products and services. • Communication channels provide stakeholders the ability to report security-related issues and to receive timely status updates on their queries. • The software vendor maintains resources to respond to reports or inquiries regarding the security of the vendor’s products and services.	Software vendors should monitor the threat landscape in order to identify new vulnerabilities and security issues that impact their software on the market. Software vendors should also provide open lines of communication to enable researchers or other stakeholders to report newly discovered vulnerabilities in the software vendor’s products and services. Communication channels could include a publicly disclosed e-mail address, website page, or other method to facilitate interactions with external researchers—for example, through a formal bug bounty program. The software vendor should also maintain teams to respond to such reports and drive processes to fix vulnerabilities in the vendor’s software. In addition to supporting the receipt of information about vulnerabilities within its software products, the software vendor should also issue communications to customers, installers, and integrators to provide information about known vulnerabilities and when fixes will be available. Fixes/patches should be developed and released in a timely manner, based on criticality and in accordance with Control Objective 4.2. Software vendor security notifications should include the criticality and potential impact of the vulnerability, as well as clear guidance for addressing the vulnerability—for example, how to install a patch or software update. Where a fix is not readily available, the software vendor should communicate the risk and provide guidance on mitigation options. Software vendor-initiated communications could include e-mail notifications, website alerts, written notices, social media posts, and any other channels the vendor maintains for stakeholder engagement. Communication channels should be publicized so that stakeholders know how to access them—for example, by signing up for e-mail notifications. Software vendor contact information should also be provided for stakeholders to submit further questions regarding security notifications.
9.2 Stakeholders are notified about security updates in a timely manner.	**9.2** The assessor shall examine evidence and interview personnel to confirm a mature process exists to notify stakeholders about security updates in a timely manner.	
9.3 Where security updates are not readily available to address known vulnerabilities or exploits, security notifications are issued to all relevant stakeholders to provide instructions for mitigating the risks associated with the known vulnerabilities and exploits.	**9.3.a** The assessor shall examine evidence and interview personnel to confirm that processes include providing stakeholders with instructions for mitigating the threat, or reducing the likelihood and/or impact of exploitation of known security issues for which a timely patch is not provided.	
	9.3.b For a sample of software security updates, examine stakeholder communications, product-specific documentation, security-testing results, or other materials to confirm that where known vulnerabilities are not addressed in the security updates, risk mitigation instructions are provided to stakeholders.	

Control Objectives	Test Requirements	Guidance
Control Objective 10: Software Update Information The software vendor provides stakeholders with detailed explanations of all software changes.		
10.1 Upon release of any software updates, a summary of the specific changes made to the software is provided to stakeholders.	**10.1.a** The assessor shall examine evidence and interview personnel to confirm the following: • A mature process exists to communicate all software changes to stakeholders upon software updates. • The process results in a clear and detailed summary of all software changes. • The change summary information clearly outlines the specific software functionality impacted by the changes. • Change details are easily accessible to stakeholders.	Release notes should be provided for all software updates, including details of any impact on software functionality and security controls. Informing stakeholders of the impact of a software update enables them to make informed decisions on whether and when to implement it.
	10.1.b For a sample of software updates, the assessor shall examine publicly available information or notifications regarding the software updates to confirm the following: • Change summary information is made available to stakeholders. • Change summary information accurately reflects the changes made to the software.	