Payment Card Industry (PCI)

Software Security Framework

Secure Software Requirements and Assessment Procedures

Version 1.1

April 2021

Document Changes

Date	Version	Description
January 2019	1.0	Initial release
April 2021	1.1	Update from v1.0. See *PCI Software Security Framework – Summary of Changes from Secure Software Requirements and Assessment Procedures Version 1.0 to 1.1* for details of changes.

Table of Contents

Introduction

To facilitate reliable and accurate payment transactions, the systems and software used as part of the payment transaction flow must be designed, developed, and maintained in a manner that protects the integrity of payment transactions and the confidentiality of all sensitive data stored, processed, or transmitted in association with payment transactions. This document, the *PCI Secure Software Requirements and Assessment Procedures* (hereafter referred to as the "PCI Secure Software Standard") provides a baseline of security requirements with corresponding assessment procedures and guidance for building secure payment software.

The *PCI Secure Software Standard* is intended for use as part of the PCI Software Security Framework. Software vendors wishing to validate their payment software under the PCI Software Security Framework would do so to this *PCI Secure Software Standard*.

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.or/document_library.

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

PCI Secure Software Requirements

The security requirements defined within the *PCI Secure Software Standard* (hereafter referred to as "PCI Secure Software Requirements") ensure that payment software is designed, engineered, developed, and maintained in a manner that protects payment transactions and data, minimizes vulnerabilities, and defends against attacks.

Scope of Requirements

The PCI Secure Software Requirements apply to the security characteristics, controls, features, and functions that payment software must possess and maintain throughout its lifecycle including, but is not limited to:

- Processes used by the software vendor to identify and support software security controls.
- Coverage of all payment software functionality, including but not limited to:
    a. End-to-end payment functionality,
    b. Inputs and outputs,
    c. Handling of error conditions,
    d. Interfaces and connections to other files, systems, and/or software or software components,
    e. All data flows, and
    f. All security mechanisms, controls, and countermeasures (e.g., authentication, authorization, validation, parameterization, segmentation, logging, etc.).
- Coverage of guidance the software vendor is expected to provide to its customers to ensure:
    a. Customers are aware how to implement and operate the payment software securely;
    b. Customers are provided guidance on configuration options of the execution environment and system components;
    c. Customers are provided guidance on how to implement security updates; and
    d. Customers and other stakeholders are aware how and where to report security issues.

  Note that the software vendor may be expected to provide such guidance even when the specific setting:

    a. Cannot be controlled by the payment software once the software is installed by the customer; or
    b. Is the responsibility of the customer and not the software vendor.
- Coverage of all supported platforms and execution environments for the payment software.
- Coverage of all tools (reporting tools, logging tools, etc.) and functions (e.g., system calls or APIs) used by or within the payment software to access critical assets.
- Coverage of all payment software components and dependencies, including supported execution platforms or environments, third-party and open-source libraries, services, and other required functions.
- Coverage of any other types of software necessary for a full implementation of the payment software.

Objective-Based Approach to Requirements

The PCI Software Security Framework has adopted an "objective-based" approach to defining the PCI Secure Software 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 software security controls and features 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 software security controls needed to meet certain requirements in this standard—for example, additional data elements identified by the software vendor as sensitive data[1]—will depend on the software vendor's risk-management priorities and processes. While this approach provides the software vendor with flexibility to implement software security controls based on identified risk, the software vendor must be able to demonstrate how the implemented controls are supported by the results of their risk-management practices. Without a robust risk-management practice in place and evidence available to support risk-based decision making, adherence to the PCI Secure Software Requirements may be difficult to validate.

Where a PCI Secure Software Requirement does not define a specific level of rigor or frequency for periodic or recurring activities—for example, the maximum period in which a software vendor must provide a security update to fix a known vulnerability—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 vendor must be able to demonstrate that its implementation provides ongoing assurance that the software security controls and security processes in place are effective and meet the intent of all applicable security requirements.

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

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

Requirements Overview

The PCI Secure Software Requirements are organized into the following three sections:

- **Secure Software Core Requirements**: General security requirements that apply to all types of payment software regardless of the software's function or underlying technology.
- **Module A – Account Data Protection Requirements**: Additional security requirements for payment software that stores, processes, or transmits account data.
- **Module B – Terminal Software Requirements**: Additional security requirements for payment software intended for deployment and operation on payment terminals (i.e., PCI-approved POI devices).

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

- **Control Objectives** – The security outcomes that must be achieved. While all control objectives must be met to be validated to this *PCI Secure Software 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 determine 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 their software, 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 and/or software vendor have 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 include software design and architecture documents (electronic or physical), source code, configuration and metadata files, and security-testing results.
- **Interview**: The assessor converses with individual personnel. The purposes 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.
- **Test**: The assessor evaluates the software code or the operation of the software using a variety of security-testing tools and techniques. At a minimum, assessors must use the appropriate combination of static and dynamic analyses to validate each control objective. Examples of such tools and techniques might include the use of automated static analysis security testing (SAST), dynamic analysis security testing (DAST), interactive application security testing (IAST), and software composition analysis (SCA) tools—as well as manual techniques such as manual code reviews and penetration testing.

The test requirements provide both software vendors and assessors with a common understanding of the expected validation activities to be performed. The specific items to be examined, observed, or tested, and the personnel to be interviewed should be appropriate for the control objective being validated and for each software vendor's unique software products and secure software lifecycle management processes. 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 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.

All test requirements are expected to be performed by the assessor. However, an assessor may choose to rely on testing performed by a third-party—including the software vendor—to satisfy a test requirement. The assessor retains full responsibility for the testing activities and results regardless of whether the testing is performed by the assessor, the software vendor, or a third-party. Where third-party testing is relied upon by the assessor, the assessor must document and justify:

- How the evidence provided by the third-party supports the same level of rigor as testing performed by the assessor, and
- How the assessor verified the evidence provided by the third-party as being appropriate for the assessor to rely on the test results.

Additionally, where software vendor-provided tests results are used, the assessor must first verify the software vendor is Secure SLC-qualified[2] before software vendor testing can be relied upon.

Sampling

Where appropriate, the assessor may utilize sampling as part of the testing process. This may include choosing representative areas of source code to examine, or a representative sample of software vendor personnel to interview. Samples must be a representative selection of the people, processes, and technologies covered by the PCI Secure Software 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 findings are 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.

Use of a Test Platform

To facilitate testing software in accordance with the assessment procedures contained in this standard, it may be necessary for the software vendor to provide a test platform. A test platform is considered to be special test functionality that is either separate or absent from production-level code. The test platform must rely on as much underlying intended production-level functionality as possible. The test platform is only to serve the purpose of providing a test framework that allows for software functionality to be exercised outside of a production-level deployment environment in order to verify the software's compliance to applicable PCI Secure Software Requirements. For example, elevated privileges or access capabilities may need to be granted for the purpose of providing run-time visibility into various facets of the software's functionality. Other examples include providing a test function to initiate a test transaction or to perform authentication functions. It is at the assessor's discretion to request any test functionality deemed required to verify the software's compliance to any applicable PCI Secure Software Requirements.

[2] Please refer to the PCI Secure SLC Standard and its associated Program Guide for more information on Secure SLC qualification.

Secure Software Core Requirements

Minimizing the Attack Surface

The attack surface of the software is minimized. Confidentiality and integrity of all software critical assets are protected, and all unnecessary features and functions are removed or disabled.

Control Objectives	Test Requirements	Guidance
Control Objective 1: Critical Asset Identification All software critical assets are identified and classified.		
1.1 All sensitive data stored, processed, or transmitted by the software is identified.	**1.1.a** The assessor shall examine vendor evidence to confirm that it details all sensitive data that is stored, processed, and/or transmitted by the software. At a minimum, this shall include all payment data; authentication credentials; cryptographic keys and related data (such as IVs and seed data for random number generators); and system configuration data (such as registry entries, platform environment variables, prompts for plaintext data in software allowing for the entry of PIN data, or configuration scripts).	Software security controls are designed and implemented to protect the confidentiality or integrity of critical assets. To make sure these controls are effective and appropriate, the software vendor should identify all sensitive data the software collects, stores, processes, or transmits, as well as all sensitive functions and resources it either provides or uses.
	1.1.b For each item of sensitive data, the assessor shall examine vendor evidence to confirm that evidence describes where this data is stored, and the applicable security controls implemented to protect the data. This includes in temporary storage (such as volatile memory), semi-permanent storage (such as RAM disks), and non-volatile storage (such as magnetic and flash storage media).	
	1.1.c The assessor shall examine vendor evidence and test the software to identify where the implementation enforces storage within a specific location or form factor (such as with an embedded system that is only capable of local storage). The assessor shall confirm that the data for all of these is supported by the vendor evidence.	

Control Objectives	Test Requirements	Guidance
	1.1.d The assessor shall examine vendor evidence and test the software to validate the information provided by the vendor in Test Requirement 1.1.a. ***Note:*** *The assessor may require and rely on assistance from the software vendor to complete this test requirement (such as through access to a dedicated test environment). Any such specific assistance must be documented by the assessor.*	
	1.1.e The assessor shall examine vendor evidence and test the software to identify the transaction types and/or card data elements that are supported by the software. The assessor shall confirm that the data for all of these is supported by the vendor evidence.	
	1.1.f The assessor shall examine vendor evidence and test the software to identify the cryptographic implementations that are supported by the software, including (but not limited to) cryptography used for storage, transport, and authentication. The assessor shall confirm that the cryptographic data for all of these implementations is supported by the vendor evidence, and that the evidence describes whether these are implemented by the software itself, through third-party software, or as functions of the execution environment.	
	1.1.g The assessor shall examine vendor evidence and test the software to identify any accounts or authentication credentials supported by the software, including both default and user created accounts. The assessor shall confirm that these accounts and credentials are supported by the vendor evidence.	
	1.1.h The assessor shall examine vendor evidence and test the software to identify any configuration options provided by the software that can impact sensitive data, including through separate files or scripts, or internal functions, menus and options provided by the software. The assessor shall confirm that these are supported by the vendor evidence.	

Control Objectives	Test Requirements	Guidance
	1.1.i When cryptography is used to protect any sensitive data, the assessor shall examine vendor evidence to confirm that these cryptographic methods and materials are identified.	
1.2 All sensitive functions and sensitive resources provided or used by the software are identified.	**1.2.a** The assessor shall examine vendor evidence to confirm that it details all sensitive functions and sensitive resources provided or used by the software. At a minimum, this shall include all functions that are designed to store, process, or transmit sensitive data, and those services, configuration files, or other information necessary for the normal and secure operation of those functions.	
	1.2.b For each of the sensitive functions and sensitive resources listed, the assessor shall examine vendor evidence to confirm that vendor evidence clearly describes how and where the sensitive data associated with these functions and resources is stored. This includes in temporary storage (such as volatile memory), semi-permanent storage (such as RAM disks), and non-volatile storage (such as magnetic and flash storage media). The assessor shall confirm that this information is supported by the information provided in Test Requirement 1.1.a.	
	1.2.c Where the sensitive functions or sensitive resources are provided by third-party software or systems, the assessor shall examine third-party software or system evidence and test the software to confirm that the vendor software is correctly following the guidance for this third-party software. ***Note:*** *For example, by reviewing the security policy of a PTS or FIPS140-2 or 140-3 approved cryptographic system.*	
	1.2.d The assessor shall examine vendor evidence and test the software to confirm that the sensitive functions and sensitive resources provided or used by the software are supported by the vendor evidence.	

Control Objectives	Test Requirements	Guidance
1.3 Critical assets are classified.	**1.3** The assessor shall examine vendor evidence to confirm that: • The vendor defines classification criteria for identifying critical assets. • Vendor classification criteria identifies the confidentiality, integrity, and resiliency requirements for each critical asset. • An inventory of all critical assets with appropriate classifications is defined.	Critical assets represent the sensitive data, functions, and resources that have business value and require confidentiality, integrity, or resiliency protection. There are numerous analysis techniques that can be used to identify critical assets, including Mission Impact Analysis (MIA), Functional Dependency Network Analysis (FDNA), and Mission Threat Analysis. Additional information and techniques can be found in publications such as the appendices of *NIST Special Publication 800-160* or in other publications from industry standards bodies such as EMVCo, ISO or ANSI.

Control Objectives	Test Requirements	Guidance
Control Objective 2: Secure Defaults Default privileges, features, and functionality are restricted to only those necessary to provide a secure default configuration.		
2.1 All functions exposed by the software are enabled by default only when and where it is a documented and justified part of the software architecture.	**2.1.a** The assessor shall examine vendor evidence and test the software to identify any software APIs or other interfaces that are provided or exposed by default upon installation, initialization, or first use. For each of these functions, the assessor shall confirm that the vendor has documented and justified its use as part of the software architecture. Testing shall include methods to reveal any exposed functionality of the software (such as scanning for listening services where applicable). ***Note:*** *This includes functions which are auto-enabled as required during operation of the software.*	Software often contains functionality (e.g., web services, administrative interface, application heartbeat, etc.) that is optional and is generally unused by many users. This functionality does not receive the same attention as standard or essential software functions and services, and often contains security weaknesses that can be exploited by malicious users to bypass security controls. To facilitate secure deployment, the software's default configuration should only expose secure functionality that has been reviewed, justified, and approved. This should include the default configuration for all software APIs, protocols, daemons, listeners, components, etc. Any unnecessary services, protocols, or ports should be disabled or removed. For guidance on services, protocols, or ports considered to be insecure, refer to industry standards and guidance (e.g., NIST, ENISA, etc.).
	2.1.b The assessor shall test the software to determine whether any of the functions identified in Test Requirement 2.1.a rely on external resources for authentication. If such resources are relied upon, the assessor shall examine vendor evidence to identify what methods are required to ensure proper authentication remains in place and shall confirm that these methods are included in the assessment of all other requirements of this standard.	
	2.1.c The assessor shall test the software to determine whether any of the functions identified in Test Requirement 2.1.a rely on external resources for the protection of sensitive data during transmission. If such resources are relied upon, the assessor shall examine vendor evidence to identify what methods are required to ensure proper protection remains in place and shall confirm that these methods are included in the assessment of all other requirements of this standard.	

Control Objectives	Test Requirements	Guidance
	2.1.d The assessor shall test the software to identify whether any of the functions identified in Test Requirement 2.1.a expose methods or services which have publicly disclosed vulnerabilities by conducting a search on the exposed protocols, methods, or services in public vulnerability repositories such as that maintained within the National Vulnerability Database.	
	2.1.e Where vulnerabilities in exposed functions exist, the assessor shall examine vendor evidence and test the software to confirm the following: • The mitigations implemented by the software vendor to minimize exploit of these weaknesses have been identified. • The risks posed by the use of known vulnerable protocols, functions, or ports is documented. • Clear and sufficient guidance on how to correctly implement sufficient security to meet the security and control objectives of this standard is made available to stakeholders per Control Objective 12.1. ***Note:*** *The assessor should reference the vendor threat information defined in Control Objective 4.1 for this item.*	
	2.1.f The assessor shall examine vendor evidence and test the software to confirm available functionality matches what is described in vendor documentation. Testing shall include methods to reveal any exposed functionality of the software (such as scanning for listening services where applicable).	

Control Objectives	Test Requirements	Guidance
	2.1.g The assessor shall examine vendor evidence for any third-party modules used by the software and ensure that any functionality exposed by each module is disabled, unable to be accessed through mitigation methods implemented by the software, or formally documented and justified by the vendor. Where access to third-party functions is prevented through implemented mitigations, the assessor shall test the software to confirm that they do not rely on a lack of knowledge of the functions as their security mitigation method—e.g., by simply not documenting an otherwise accessible API interface—and to verify the mitigations in place are effective at preventing the insecure use of such third-party functions.	
2.2 All software security controls, features, and functions are enabled upon software installation, initialization, or first use. ***Note:*** *Specific software security controls required to protect the integrity and confidentiality of sensitive data, sensitive functions, and sensitive resources are captured in the Software Protection Mechanisms section.*	**2.2.a** The assessor shall examine vendor evidence and test the software to identify all software security controls, features and functions, and to confirm that any such controls, features and functions relied upon by the software for the protection of critical assets are enabled upon installation, initialization, or first use of the software.	As previously noted earlier in guidance, software security controls are designed and implemented to protect the confidentiality and integrity of critical assets. Examples of such software security controls include authentication and authorization mechanisms, cryptographic controls, and controls to prevent leakage of sensitive data. Default software settings should result in a secure software configuration and should not rely on the end-user being a subject-matter expert to facilitate a secure configuration. To that effect, all available software security controls should be active upon software installation, initialization, or first use, depending upon how the software is deployed.
	2.2.b Where any software security controls, features and functions are enabled only upon initialization or first use, the assessor shall test the software to confirm that no sensitive data can be processed until this initialization process has been completed.	
	2.2.c Where user input or interaction is required to enable any software security controls, features, or functions (such as the installation of certificates) the assessor shall examine vendor evidence to confirm that there is clear and sufficient guidance on the process provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	

Control Objectives	Test Requirements	Guidance
	2.2.d The assessor shall examine vendor evidence and test the software to confirm that following the software vendor's security guidance required in Control Objective 12.1 results in all security-relevant software security controls, features, and functions being enabled prior to the software enabling processing of sensitive data.	
2.3 Default authentication credentials or keys for built-in accounts are not used after installation, initialization, or first use.	**2.3.a** The assessor shall examine vendor evidence to identify all default credentials, keys, certificates, and other critical assets used for authentication by the software. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objectives 1, 5, and 7 to determine the authentication and access control mechanisms, keys, and other critical assets used for authentication.*	To protect against unauthorized access, payment software should prevent the use of built-in accounts until the default authentication credentials can be changed. Built-in accounts with known credentials such as default or empty passwords, or default keys are often overlooked during installation, initial configuration, or use, and can be used by a malicious user to bypass access controls. Therefore, the software should not use or rely on the default credentials for its operation upon installation, initialization, or first use.
	2.3.b The assessor shall test the software to confirm that all default credentials, keys, certificates, and other critical assets used for authentication by the software are supported by the vendor evidence. ***Note:*** *It is expected that this analysis will include, but not necessarily be limited to, the use of entropy analysis tools to look for hardcoded cryptographic keys, searches for common cryptographic function call and structures such as S-Boxes and big-number library functions (and tracing these functions backwards to search for hardcoded keys), as well as checking for strings containing common user account names or password values.*	

Control Objectives	Test Requirements	Guidance
	2.3.c Where user input or interaction is required to disable or change any authentication credentials or keys for built-in accounts, the assessor shall examine vendor evidence to confirm that there is clear and sufficient guidance on this process provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	2.3.d The assessor shall test the software to confirm that default authentication credentials or keys for built-in accounts are not used by the authentication and access control mechanisms implemented by the software after software installation, initialization, or first use. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 5 to determine the authentication and access control mechanisms implemented by the software.*	
	2.3.e The assessor shall test the software to confirm that default authentication credentials or keys for built-in accounts are not used to protect the storage and transmission of sensitive data. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 6 to determine the software security controls implemented to protect sensitive data.*	

Control Objectives	Test Requirements	Guidance
2.4 The privileges and resources requested by the software from its execution environment are limited to those necessary for the operation of the software.	**2.4.a** The assessor shall examine vendor evidence to identify all privileges and resources required by the software and to confirm the evidence describes and reasonably justifies all privileges and resources required, including explicit permissions for access to resources, such as cameras, contacts, etc.	In many attacks on software or underlying systems, the software is often used to execute functions on the underlying operating systems or to abuse accessible external resources. When the software requires excessive permissions, those permissions could be exploited by a malicious user. To minimize the software's attack surface, the software should only request and be granted the minimum required privileges for its intended operation. For example, system service accounts that the software uses to operate, or accounts used by the software to access underlying components such as a database or invoke web-services calls should not require permissions that exceed the minimum necessary for the software to perform its operations. The same concept applies to resources used by the software. The software should be granted access to only the minimum required resources for its expected operation. For example, mobile applications that do not require access to the camera or photographs should not request such access unless they are a necessary part of the software architecture. Similarly, software should not have access to sensitive files (e.g., /etc/passwd or Ntuser.dat) unless there is a legitimate need for the software to access those files.
	2.4.b Where limiting access is not possible—e.g., due to the architecture of the solution or the execution environment in which the software is executed—the assessor shall examine vendor evidence to identify all mechanisms implemented by the software to prevent unauthorized access, exposure, or modification of critical assets, and to confirm there is clear and sufficient guidance on properly implementing the mechanisms provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	2.4.c The assessor shall test the software to confirm that access permissions and privileges are assigned according to the vendor evidence. The assessor shall, where possible, use suitable tools for the platform on which the software is installed to review the permissions and privileges of the software itself, as well as the permissions and privileges of any resources, files, or additional elements generated or loaded by the software during use. ***Note:*** *Where the above testing is not possible, the assessor shall justify why this is the case and that the testing that has been performed is sufficient.*	
	2.4.d Where the software execution environment provides legacy features for use by older versions of the software, the assessor shall examine vendor evidence and test the software to confirm that these are not utilized, and only recent and secured functionality is implemented. For example, software should "target" the latest versions of APIs provided by the environment they run on, where available.	

Control Objectives	Test Requirements	Guidance
2.5 Default privileges for built-in accounts are limited to those necessary for their intended purpose and function.	**2.5.a** The assessor shall examine the vendor evidence to identify all default accounts provided by the software and to confirm vendor evidence includes reasonable justification for the privileges assigned to these accounts.	In support of the principle of "least privilege," built-in accounts should only have the privileges required for the intended function of the account, including access to sensitive data and resources as well as the ability to execute sensitive functions. For example, a built-in administrator account may require the ability to configure the software and associated user accounts, but not the ability to access areas containing sensitive data. Applying the principle of least privilege to user accounts helps prevent users without sufficient knowledge about the software from incorrectly or accidentally changing the software configuration or its security settings. Enforcing least privilege also helps to minimize the effects of unauthorized access to software user accounts. To limit access to sensitive data, functions, and resources to only those accounts that require such access, the level of privilege and access required should be defined and documented for each built-in account—e.g., an access matrix—such that its assigned functions may be performed, but that no additional or unnecessary access or privileges are granted.
	2.5.b The assessor shall test the software to confirm that all default accounts provided or used by the software are supported by the vendor evidence.	
	2.5.c The assessor shall examine vendor evidence and test the software to confirm that exposed functionalities (i.e., APIs) are protected from use by unauthorized users to modify account privileges and elevate user access rights.	

Control Objectives	Test Requirements	Guidance
Control Objective 3: Sensitive Data Retention Retention of sensitive data is minimized.		
3.1 The software only retains the sensitive data absolutely necessary for the software to provide its intended functionality.	**3.1.a** The assessor shall examine vendor evidence to identify what sensitive data is collected by the software for use beyond any one transaction, the default time period for which it is retained, and whether the retention period is user-configurable, and to confirm vendor evidence includes reasonable justification for retaining the sensitive data. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.1 to determine the sensitive data retained by the software.*	To prevent the unauthorized disclosure of sensitive data to unauthorized parties, the software should retain sensitive data only for the duration necessary to perform the specific operation for which sensitive data is collected. Retaining sensitive data longer than required presents opportunity for the data to be mishandled, misused, or accidentally disclosed. This control objective differentiates between transient sensitive data retained temporarily to facilitate software operation (e.g., retention of payment information in memory until completion of the authorization process) and sensitive data that is retained on a more permanent basis for the intended business use when the software user configures the retention period.
	3.1.b The assessor shall test the software to confirm that all available functions or services designed for the retention of sensitive data are supported by the vendor evidence. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.2 to determine the sensitive functions and services provided or used by the software.*	
	3.1.c The assessor shall test the software to confirm that sensitive data stored solely for the purposes of debugging, error finding, or testing of systems is protected during storage in accordance with Control Objective 6. Any such functionality that allows for storage of sensitive data must be explicitly enabled through an interface that requires interaction and authorization by the user and retained only for the duration necessary in accordance with reasonable vendor criteria. Closure of the software must result in termination of this debugging state, such that it requires explicit re-enablement when the software is next executed; and any sensitive data is securely deleted per Control Objective 3.4.	

Control Objectives	Test Requirements	Guidance
	3.1.d Where user input or interaction is required to configure the retention period of sensitive data, the assessor shall examine vendor evidence to confirm that there is clear and sufficient guidance on this process provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
3.2 Transient sensitive data is retained only for the duration necessary to fulfill a legitimate business purpose.	**3.2.a** The assessor shall examine vendor evidence to identify all sensitive data that is retained by the software for transient use, what triggers the secure deletion of this data, and confirm reasonable justification exists for retaining the data. This includes data that is stored only in memory during the operation of the software. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.1 to determine the transient sensitive data retained temporarily by the software.*	Sensitive data elements collected in conjunction with software operations should only be retained for as long as required to complete that operation or related transaction. After payment processing is complete, all transient sensitive data should be securely deleted from all locations where it has been retained such that any subsequent process, component, function, application, entity, etc., within the environment may not access or capture the sensitive data. Software vendors should also be aware of and account for how other aspects of the software architecture (such as the software-development language and operating environment) may affect how and where transient sensitive data is retained. For example, operating-system usage of swap partitions or virtual memory files can cause information that should have been transient to persist longer than intended. *(continued on next page)*
	3.2.b The assessor shall test the software to confirm that all available functions or services that retain transient sensitive data are supported by vendor evidence and do not use immutable objects. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.2 to determine the sensitive functions and services that retain transient sensitive data.*	

Control Objectives	Test Requirements	Guidance
	3.2.c The assessor shall test the software to confirm that transient sensitive data stored solely for the purposes of debugging, error finding, or testing of systems is protected in accordance with Control Objective 6. Any such functionality that allows for the storage of transient sensitive data must be explicitly enabled through an interface that requires interaction and authorization by the user. Closure of the software must result in termination of this debugging state, such that it requires explicit re-enablement when the software is next executed; and any transient sensitive data is securely deleted in accordance with Control Objective 3.4.	If any sensitive data (e.g., pre-authorization data, etc.) must be used for debugging or troubleshooting purposes, the software should only capture the minimum amount of data necessary and store it securely in a known location.
	3.2.d Where users can configure retention of transient sensitive data, the assessor shall examine vendor evidence to confirm that clear and sufficient guidance on this process is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
3.3 The software protects the confidentiality and integrity of sensitive data (both transient and persistent) during retention. ***Note:*** *The Software Protection Mechanisms section includes several specific software security controls that are required to be implemented to protect sensitive data during storage, processing or transmission. Those software security controls should be analyzed to determine their applicability to the types of sensitive data retained by the software.*	**3.3.a** The assessor shall examine the vendor evidence to identify the protection methods implemented for all sensitive data during storage and transmission.	The software should maintain security controls and mechanisms to protect all sensitive data while it is retained by the software. Examples of software security controls include writing to a secure memory location or using cryptography to render the data unreadable.
	3.3.b The assessor shall test the software to confirm that no additional storage of sensitive data is included.	
	3.3.c Where sensitive data is stored outside of temporary variables within the code itself, the assessor shall test the software to confirm that sensitive data is protected using either strong cryptography or other methods that provide an equivalent level of security.	

Control Objectives	Test Requirements	Guidance
	3.3.d Where protection methods use cryptography, the assessor shall examine vendor evidence and test the software to confirm that the cryptographic implementation complies with Control Objective 7 of this standard.	
	3.3.e Where sensitive data is protected using methods other than strong cryptography, the assessor shall examine vendor evidence and test the software to confirm that the protections are present in all environments where the software is designed to be executed, are correctly implemented, and are covered by the vendor evidence.	
	3.3.f Where users are required to configure protection methods, the assessor shall examine vendor evidence to confirm that there is clear and sufficient guidance on this process provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
3.4 The software securely deletes sensitive data when it is no longer required.	**3.4.a** The assessor shall examine vendor evidence to identify all secure deletion methods implemented by the software for all non-transient sensitive data.	Secure deletion may be required at the end of a software-specific operation or upon completion of user-specified retention requirements. In the latter case, the software should be developed to provide functionality to facilitate secure deletion of the sensitive data after expiry of the user-specified retention period. Only in circumstances where the retention of sensitive data is explicitly permitted should the data be retained after transaction processing is complete.
	3.4.b The assessor shall examine vendor evidence and test the software to identify any platform or implementation level issues that complicate the secure deletion of non-transient sensitive data and to confirm that any non-transient sensitive data is securely deleted using a method that ensures that the data is unrecoverable after deletion. Methods may include (but are not necessarily limited to) overwriting the data, deletion of cryptographic keys (of sufficient strength) which have been used to encrypt the data, or platform specific functions which provide for secure deletion. Methods must accommodate for platform specific issues, such as flash wear-leveling algorithms or SSD over-provisioning, which may complicate simple over-writing methods.	

Control Objectives	Test Requirements	Guidance
	3.4.c The assessor shall test the software using forensic tools to identify any non-transient sensitive data residue in the execution environment, and to confirm that the methods attested by the software vendor are correctly implemented and applied to all sensitive data. This analysis should accommodate for the data structures and methods used to store the sensitive data (e.g., by examining file systems at the allocation level, and translating data formats to identify sensitive data elements), as well as covering all non-transient sensitive data types. ***Note:*** *Where forensic testing of some or all aspects of the platform is not possible, the assessor should examine additional evidence to confirm secure deletion of sensitive data. Such evidence may include (but is not necessarily limited to) memory and storage dumps from development systems, evidence from memory traces from emulated systems, or evidence from physical extraction of data performed on-site by the software vendor.*	
3.5 Transient sensitive data is securely deleted from temporary storage facilities automatically by the software once the purpose for which it is retained is satisfied.	**3.5.a** The assessor shall examine vendor evidence to identify all secure deletion methods for all transient sensitive data and to confirm that these methods ensure that the data is unrecoverable after deletion. ***Note:*** *This includes data which may be stored only temporarily in program memory / variables during operation of the software.*	Where sensitive data is only retained temporarily to perform a specific function (such as a payment transaction), mechanisms are required to securely delete the sensitive data once the specific function has completed. Transient sensitive data is often erased from temporary storage locations after processing is complete. However, that data may remain resident in volatile memory (RAM) or in other storage locations for longer periods than anticipated (such as in swap files/partitions or log files). Software vendors should account for all locations where sensitive data is stored, regardless of the intended duration of storage, and ensure that such data is securely deleted once the purpose for which the software collected the data has been satisfied.
	3.5.b The assessor shall examine vendor evidence and test the software to identify any platform or implementation level issues that complicate the erasure of such transient sensitive data—such as abstraction layers between the code and the hardware execution environment—and to confirm what methods have been implemented to minimize the risk posed by these complications.	

Control Objectives	Test Requirements	Guidance
	3.5.c The assessor shall test the software, including usage of forensic tools, to identify any sensitive data residue in the execution environment to confirm that the methods attested by the software vendor are correctly implemented and applied to all transient sensitive data. This analysis should accommodate for the data structures and methods used to store the sensitive data—e.g., by examining file systems at the allocation level, and translating data formats to identify sensitive data elements—as well as cover all non-transient sensitive data types. ***Note:*** *Where forensic testing of some or all aspects of the platform is not possible, the assessor should examine additional evidence to confirm secure deletion of sensitive data. Such evidence may include (but is not necessarily limited to) memory and storage dumps from development systems, evidence from memory traces from emulated systems, or evidence from physical extraction of data performed on-site by the software vendor.*	

Control Objectives	Test Requirements	Guidance
3.6 The software does not disclose sensitive data through unintended channels.	**3.6.a** The assessor shall examine vendor evidence to confirm the software vendor has performed a thorough analysis to account for all sensitive data disclosure attack vectors including, but not limited to: • Error messages, error logs, or memory dumps. • Execution environments that may be vulnerable to remote side-channel attacks to expose sensitive data—such as attacks that exploit cache timing or branch prediction within the platform processor. • Automatic storage or exposure of sensitive data by the underlying execution environment, such as through swap-files, system error logging, keyboard spelling, and auto-correct features, etc. • Sensors or services provided by the execution environment that may be used to extract or leak sensitive data such as through use of an accelerometer to capture input of a passphrase to be used as a seed for a cryptographic key, or through capture of sensitive data through use of cameras, near-field communication (NFC) interfaces, etc.	Proactive measures to ensure that sensitive data is not inadvertently "leaked" should be implemented by the software vendor or within the software. Disclosure of sensitive data to unauthorized parties often occurs via unknown or unintended outputs or channels. For example: sensitive data could be unintentionally disclosed through error- or exception-handling routines, logging or debugging channels, third-party services and/or components, or through the use of shared resources such as memory, disk, files, keyboards, displays, and functions. Protective mechanisms, whether process or programmatic in nature, should be implemented to ensure that sensitive data is not accidentally disclosed through such means.
	3.6.b The assessor shall examine vendor evidence, including the results of the analysis described in Test Requirement 3.6.a, and test the software to confirm the software vendor implements mitigations to protect against unintended disclosure of sensitive data. Mitigations may include usage of cryptography to protect the data, or the use of blinding or masking of cryptographic operations (where supported by the execution environment).	
	3.6.c The assessor shall examine vendor evidence to confirm that clear and sufficient guidance on the proper configuration and use of such mitigations is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	

Control Objectives	Test Requirements	Guidance
	3.6.d The assessor shall test the software using forensic tools to identify any sensitive data residue in the execution environment, and to confirm that all mitigation controls are correctly implemented and the software does not expose or otherwise reveal sensitive data.	

Software Protection Mechanisms

Software security controls are implemented to protect the integrity and confidentiality of critical assets.

Control Objectives	Test Requirements	Guidance
Control Objective 4: Critical Asset Protection Critical assets are protected from attack scenarios.		
4.1 Attack scenarios applicable to the software are identified. ***Note:*** *This control objective is an extension of Control Objective 10.1. Validation of both control objectives should be performed at the same time.*	**4.1.a** The assessor shall examine vendor evidence to confirm that the software vendor has identified, documented, and prepared mitigations for relevant attack scenarios for the software.	Software vendors should evaluate the design of their payment software to identify attack scenarios applicable to the software, and the results of that analysis should be documented. Documentation should describe the various aspects of the code that could be attacked (including tasks or actions that frameworks and libraries do on the software's behalf), the difficulty in mounting a successful attack, how widely the program will be distributed, and what mitigation techniques are used (for example, how the security functionality of the operating system is leveraged) and identify or define a methodology for measuring the likelihood and impact of an exploit. When the software relies on execution environment security controls, the software vendor should review and reference the implementation documentation for the platform—such as Security Policies for PCI-approved POI devices or FIPS140-2 or 140-3 approved cryptographic modules—and confirm that the software and its associated documentation correctly and completely accommodates for the guidance in these documents.
	4.1.b The assessor shall examine vendor evidence to determine whether any specific industry-standard methods or guidelines were used to identify relevant attack scenarios, such as the threat model guidelines. Where such industry standards are not used, the assessor shall confirm that the methodology used provides an equivalent coverage of the attack scenarios and methods for the software.	
	4.1.c The assessor shall examine the vendor evidence to confirm the following: • A formal owner of the software is assigned. This may be a role for a specific individual or a specific name, but evidence must clearly show an individual who is accountable for the security of the software. • A methodology is defined for measuring the likelihood and impact for any exploit of the system. • Generic threat methods and types that may be applicable to the software are documented. • All critical assets managed by and all sensitive resources used by the system are documented. *(continued on next page)*	

Control Objectives	Test Requirements	Guidance
	4.1.c • All entry and egress methods for sensitive data by the software, as well as the authentication and trust model applied to each of these entry/egress points, are defined. • All data flows, network segments, and authentication/privilege boundaries are defined. • All static IPs, domains, URLs, or ports required by the software for operation are documented. • Considerations for cryptography elements like cipher modes, protecting against timing attacks, padded oracles, brute force, “rainbow table” attacks, dictionary attacks against the input domain, etc. are documented. • Execution environment implementation specifics or assumptions such as network configurations, operating system security configurations, etc. are documented. • Consideration for the installed environment of the software, including any considerations for the size of the install base are documented. All attack surfaces that must be mitigated—such as implementing insecure user prompts or separating open protocol stacks; storage of sensitive data post authorization or storage of sensitive data using insecure methods, etc.—are documented.	
	4.1.d The assessor shall examine vendor evidence to confirm that the threat model created is reasonable to address the potential risks posed by the install and use of the software in a production environment—i.e., not in a test environment—given the assessor’s understanding through evaluation of the payment software to this standard.	

Control Objectives	Test Requirements	Guidance
4.2 Software security controls are implemented to mitigate software attacks.	**4.2.a** The assessor shall examine vendor evidence to confirm that for each of the threats identified in Control Objective 4.1, one or more mitigation methods are clearly defined, or reasonable justification for the lack of mitigations is provided.	Once attack scenarios are identified, the risk of their occurrence should be mitigated. Software vendors should define and implement mechanisms to protect the software from attacks and reduce the likelihood and impact of successful execution. Any attack scenarios left unmitigated or insufficiently mitigated should be reasonably justified. The exact nature of the protection mechanism(s) will depend on the attack scenarios, the development platform, the software-development language, frameworks, libraries and APIs used by the software, as well as the operating environment (e.g., mobile device or distributed cloud-based application) upon which the software is intended to be deployed. To minimize the attack surface of the software, the software can be developed using secure design principles such as layered defense, application segmentation and isolation (logical), and adaptive response. Examples of software security controls include input and output validation, authentication, parameterization, escaping, segmentation, logging, etc. For guidance on implementing cyber resiliency techniques and approaches, refer to industry standards and guidance such as *NIST Special Publication 800-160,* Appendix E.
	4.2.b The assessor shall examine vendor evidence and test the software to confirm that the implemented mitigation methods are reasonable for the threat they address.	
	4.2.c Where any mitigations rely on settings within the software, the assessor shall test the software to confirm that such settings are applied by default, before first processing any sensitive data, upon installation, initialization, or first use of the software. Where user input or interaction can disable, remove, or bypass any such mitigations, the assessor shall test the software to confirm that such action requires authorization and strong authentication, and examine vendor evidence to confirm that clear and sufficient guidance on the risk of this action and that installation in this manner will invalidate any security validation that has been performed is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	4.2.d When any mitigations rely on features of the execution environment, the assessor shall examine vendor evidence to confirm that guidance is provided to the software users to enable such settings as part of the install process. Where the execution environment provides APIs to query the status of mitigation controls, the assessor shall test the software to confirm that software checks for these mitigations are in place and active prior to being launched, and periodically throughout execution.	

Control Objectives	Test Requirements	Guidance
Control Objective 5: Authentication and Access Control The software implements strong authentication and access control to help protect the confidentiality and integrity of critical assets.		
5.1 Access to critical assets is authenticated.	**5.1.a** The assessor shall examine vendor evidence to confirm that the vendor has identified authentication requirements (i.e., type and number of factors) for all roles based on critical asset classification, the type of access (e.g., local, non-console, remote) and level of privilege. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.3 to determine the classifications for all critical assets.*	Secure authentication facilitates individual responsibility for actions and allows the software to maintain an effective audit trail of user activity. This expedites issue resolution and containment when misuse or malicious intent occurs. Authentication mechanisms should cover all non-public resources managed by or accessible through the software, as well as sensitive functions that can alter the software functionality or impact the security of the sensitive data and resources. Examples of authentication methods include: • Something you know, such as a password or passphrase • Something you have, such as a token device or smart card • Something you are, such as a biometric To ensure that the implemented authentication mechanisms are adequate to address the risk of unauthorized access to sensitive data or sensitive resources, or misuse of a sensitive function, the vendor should analyze threats and identify the required level of authentication for all types of users and roles. *(continued on next page)*
	5.1.b The assessor shall examine vendor evidence and test the software to confirm that all access to critical assets is authenticated and authentication mechanisms are implemented correctly.	
	5.1.c Where the software recommends, suggests, relies on, or otherwise facilitates the use of additional mechanisms (such as third-party VPNs, remote desktop features, etc.) to facilitate secure non-console access to the system on which the software is executed or directly to the software itself, the assessor shall examine vendor evidence to confirm that clear and sufficient guidance on how to configure authentication mechanisms correctly is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	

Control Objectives	Test Requirements	Guidance
	5.1.d The assessor shall examine vendor evidence to confirm that any sensitive data associated with credentials, including public keys, is identified as a critical asset.	For example, a user with limited access to sensitive data and sensitive resources could be required to perform authentication using a single authentication factor (e.g., password or a passphrase) while a user that is able to export the entire database might be required to perform multi-factor authentication. Other factors such as type of access (e.g., local, non-console, remote) and level of privilege (e.g., the ability to invoke sensitive functions such as pause logging or change access privileges) may influence the level of authentication that should be required. The software should not require the use of any group, shared, or generic accounts. The use of group or shared accounts makes it more difficult to determine which individuals execute specific actions since a given action could have been performed by anyone that has knowledge of the group or shared accounts' authentication credentials. ***Note:*** *Control Objectives 3.3 and 4.2 require sensitive data, sensitive functions, and sensitive resources to be protected. Control Objectives 1.1 and 1.2 require sensitive data, sensitive functions, and sensitive resources to be identified/defined.*
5.2 Access to critical assets requires unique identification.	**5.2.a** The assessor shall examine vendor evidence and test the software to confirm that all implemented authentication methods require unique identification.	
	5.2.b Where interfaces, such as APIs, allow for automated access to critical assets, the assessor shall examine vendor evidence and test the software to confirm that unique identification of different programs or systems accessing the critical assets is required (for example, through use of multiple public keys) and that guidance on configuring a unique credential for each program or system is included in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	5.2.c Where identification is supplied across a non-console interface, the assessor shall test the software to confirm that authentication mechanisms are protected.	
	5.2.d The assessor shall examine vendor evidence to confirm that the software vendor's implementation guidance provided to stakeholders per Control Objective 12.1 specifically notes that identification and authentication parameters must not be shared between individuals, programs, or in any way that prevents the unique identification of each access to a critical asset.	
	5.2.e The assessor shall examine vendor evidence, including source code of the software, to confirm that there are no additional methods for accessing critical assets.	

Control Objectives	Test Requirements	Guidance
5.3 Authentication methods (including session credentials) are sufficiently strong and robust to protect authentication credentials from being forged, spoofed, leaked, guessed, or circumvented.	**5.3.a** The assessor shall examine vendor evidence to confirm that all implemented authentication methods were evaluated to identify the details of known vulnerabilities or attack methods on the authentication method, and how the implementation mitigates against such attacks. The evidence must also illustrate that the implementation used in the software was considered. For example, a fingerprint may be uniquely identifiable to an individual, but the ability to spoof or otherwise bypass such technology can be highly dependent on the way the solution is implemented. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 4.1 to determine the attack scenarios applicable to the software.*	The software vendor must evaluate, document, and justify the usage of implemented authentication methods to demonstrate that they are sufficiently strong to protect authentication credentials in the software's intended specific use case or deployment scenario. For example, if the software uses biometric authentication, the vendor may want to identify all points at which a malicious user may attack the authenticator and implement mitigations to address those risks. The authentication mechanism implemented in the software could rely on additional sensors to ensure the provided biometric sample is from a living human and not a forged or spoofed sample. In some use cases or deployment scenarios, an authentication mechanism that relies on a single authentication method may not be sufficient. In such circumstances, the software vendor may want to implement additional mitigation strategies (e.g., multi-factor authentication mechanism). To support a claim that the implemented authentication mechanism is sufficiently strong and robust, a vendor should adopt an industry-accepted methodology for assigning assurance levels (e.g., *NIST SP800-63-3* and *NIST SP800-63B*).
	5.3.b The assessor shall examine vendor evidence to confirm that implemented authentication methods are robust and that robustness of the authentication methods was evaluated using industry-accepted methods. ***Note:*** *The vendor assessment and robustness justification include consideration of the full path of the user credentials, from any input source (such as a Human Machine Interface or other program), through transition to the execution environment of the software (including any switched/network transmissions and traversal through the execution environment's software stack before being processed by the software itself).*	
	5.3.c The assessor shall test the software to confirm the authentication methods are implemented correctly and do not expose vulnerabilities.	

Control Objectives	Test Requirements	Guidance
5.4 By default, all access to critical assets is restricted to only those accounts and services that require such access.	**5.4.a** The assessor shall examine vendor evidence to confirm that the vendor has clearly identified and reasonably justified the required access for all critical assets.	To ensure the software protects the confidentiality and integrity of critical assets, access privileges to those critical assets should be restricted based on vendor-defined access requirements. There are various approaches to implementing privilege restriction, such as trust-based privilege management, attribute-based usage restriction, and dynamic privileges. To reduce the attack surface of the software, the software authorization mechanisms might limit access to critical assets to only those accounts that need such access—i.e., the principle of "least privilege." Other techniques include implementation of Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), time-based adjustment to privilege, and dynamic revocation of access authorization.
	5.4.b The assessor shall examine vendor evidence and test the software to identify what access is provided to critical assets and confirm that such access correlates with the vendor evidence. The test to confirm access is restricted should include attempts to access critical assets through user accounts, roles, or services which should not have the required privileges.	

Control Objectives	Test Requirements	Guidance
Control Objective 6: Sensitive Data Protection Sensitive data is protected at rest and in transit.		
6.1 Sensitive data is secured anywhere it is stored.	**6.1.a** The assessor shall examine vendor evidence and test the software to identify all locations where sensitive data is stored to confirm protection requirements for all sensitive data are defined, including requirements for rendering sensitive data with confidentiality considerations unreadable anywhere it is stored persistently.	Sensitive data must be protected wherever it is stored. In some cases, the integrity may be the primary concern. In other cases, it may be the confidentiality of the sensitive data that must be protected. Sometimes, both the integrity and confidentiality must be secured. The type of data and the purpose for which it is generated will often determine the need for integrity or confidentiality protection. In all cases, those protection requirements must be clearly defined. In cases where the confidentiality of sensitive data is a concern, it is imperative to know where and for how long the data is retained. The vendor must have details of all locations where the software may store the data, including in any underlying software or systems—such as OS, log files, databases, etc.—as well as documentation of the security controls used to protect the data.
	6.1.b The assessor shall examine vendor evidence and test the software to confirm that security methods implemented to protect all sensitive data during storage appropriately address all defined protection requirements and identified attack scenarios. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.1 to determine all sensitive data retained by the software, and Control Objective 4.1 to identify all attack scenarios applicable to the software.*	
	6.1.c Where cryptography is used for securing sensitive data, the assessor shall examine vendor evidence and test the software to confirm that any method implementing cryptography for securing sensitive data complies with Control Objective 7.	
	6.1.d Where index tokens are used for securing sensitive data, the assessor shall examine vendor evidence and test the software to confirm that these are generated in a way that ensures there is no correlation between the value and the sensitive data that is being referenced (without access to the vendor software to perform the correlation as part of a formally defined and assessed feature of that software—such as "de-tokenization").	Sensitive data requiring confidentiality protection, when stored persistently, must be protected to prevent malicious or accidental access. Examples of methods to render sensitive data unreadable include usage of a one-way hash or the use of strong cryptography with associated key-management processes. (Refer to Control Objective 7 for more information and guidance on strong cryptography.) Other approaches might involve the use of an index token or a one-time pad. An index token is a cryptographic token that replaces the sensitive data based on a given index for an unpredictable value. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	6.1.e Where protection methods rely on security properties of the execution environment, the assessor shall examine vendor evidence and test the software to confirm that these security properties are valid for all platforms which the software targets, and that they provide sufficient protection to the sensitive data.	A one-time pad is a system in which a randomly generated private key is used only once to encrypt a message that is then decrypted using a matching, one-time pad and key. Where the integrity of sensitive data is a concern, strong cryptography with appropriate key-management practices is one method that could be used to satisfy integrity protection requirements during storage.
	6.1.f Where protection methods rely on security properties of third-party software, the assessor shall examine vendor evidence and test the software to confirm that this software provides security that is sufficient to meet the requirements of this standard. The assessor shall perform a review of current publicly available literature and vulnerability disclosures to confirm that there are no unmitigated vulnerabilities or issues with the security properties relied upon with that software.	
6.2 Sensitive data is secured during transmission.	**6.2.a** The assessor shall examine vendor evidence and test the software to identify all locations within the software where sensitive data is transmitted and confirm protection requirements for the transmission of all sensitive data are defined.	To prevent malicious individuals from intercepting or diverting sensitive data while in transit, it must be protected during transmission. One method to protect sensitive data in transit is to encrypt it using strong cryptography prior to transmission. (Refer to Control Objective 7 for more information and guidance on strong cryptography.) Alternatively, the software could establish an authenticated and encrypted channel using only trusted keys and certificates (for authentication) and appropriate encryption strength for the selected protocols.
	6.2.b The assessor shall examine vendor evidence and test the software to confirm that for each of the ingress and egress methods that allow for transmission of sensitive data with confidentiality considerations outside of the physical execution environment, sensitive data is always encrypted with strong cryptography prior to transmission or is transmitted over an encrypted channel using strong cryptography. ***Note:*** *The assessor should refer to evidence obtained in the testing of Control Objective 1.1 to determine the sensitive data stored, processed or transmitted by the software.*	
	6.2.c Where third-party or execution-environment features are relied upon for the security of the transmitted data, the assessor shall examine vendor evidence to confirm that clear and sufficient guidance on how to configure such features are included in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	

Control Objectives	Test Requirements	Guidance
	6.2.d Where transport layer encryption is used to secure the transmission of sensitive data, the assessor shall test the software to confirm that all ingress and egress methods enforce a secure version of the protocol with end-point authentication prior to the transmission of that sensitive data.	
	6.2.e Where the methods implemented for encrypting sensitive data allow for the use of different types of cryptography or different levels of security, the assessor shall test the software, including capturing software transmissions, to confirm the software enforces the use of strong cryptography at all times during transmission.	
6.3 Use of cryptography meets all applicable cryptography requirements within this standard.	**6.3.a** The assessor shall examine vendor evidence and test the software to confirm that each use of cryptography—where cryptography is relied upon (in whole or in part) for the security of critical assets—is compliant to Control Objective 7. ***Note:*** *The assessor should refer to Control Objective 7 to identify all requirements for appropriate and correct implementation of cryptography.*	Wherever cryptography is used to meet software-security requirements in this standard, it must be done in accordance with the specific security requirements related to the use of cryptography (including those in Control Objective 7). For example, storing a cryptographic key (used for protecting sensitive data) in a plaintext file would not be considered sufficient security unless additional controls prevented the file containing the cryptographic key from being accessed, modified, or exposed. Further guidance on appropriate uses of cryptographic algorithms can be found in current versions of *NIST SP 800-175* or in other related industry guidance from ISO or ANSI.
	6.3.b Where cryptographic methods provided by third-party software or aspects of the execution environment or platform on which the application is run are relied upon for the protection of sensitive data, the assessor shall examine vendor evidence and test the software to confirm that the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1 provides clear and sufficient detail for correctly configuring these methods during the installation, initialization, or first use of the software.	
	6.3.c Where asymmetric cryptography such as RSA or ECC is used for protecting the confidentiality of sensitive data, the assessor shall examine vendor evidence and test the software to confirm that private keys are not used for providing confidentiality protection to the data.	

Control Objectives	Test Requirements	Guidance
Control Objective 7: Use of Cryptography Cryptography is used appropriately and correctly.		
7.1 Approved cryptographic algorithms and methods are used for securing critical assets. Approved cryptographic algorithms and methods are those recognized by industry-accepted standards bodies—for example: NIST, ANSI, ISO, and EMVCo. Cryptographic algorithms and parameters that are known to be vulnerable are not used.	**7.1.a** The assessor shall examine the vendor evidence to confirm that, where cryptography is relied upon (in whole or in part) for the security of the critical assets: • Industry-accepted cryptographic algorithms and modes of operation are used in the software as the primary means for protecting critical assets; and • Use of any unapproved algorithms must be in conjunction with approved algorithms and implemented in a manner that does not reduce the equivalent cryptographic key strength provided by the approved algorithms.	Not all cryptographic algorithms are sufficient to protect sensitive data. It is a well-established principle in software security to utilize only recognized cryptographic implementations based on current, industry-accepted standards such as those from industry bodies like NIST, ANSI, ISO, and EMVCo. The use of proprietary cryptographic implementations may increase the risk of data compromise as proprietary implementations are often not subjected to the same rigorous level of testing that industry-accepted implementations have undergone. Only those implementations that have been subjected to sufficient testing (for example, by NIST, ANSI, or other recognized industry bodies) should be used.
	7.1.b The assessor shall examine vendor evidence, including the vendor threat information, and test the software to confirm that: • Only documented cryptographic algorithms and modes are used in the software and are implemented correctly, and • Protections are incorporated to prevent common cryptographic attacks such as the use of the software as a decryption oracle, brute-force or dictionary attacks against the input domain of the sensitive data, the re-use of security parameters such as IVs, or the re-encryption of multiple datasets using linearly applied key values (such as XOR'd key values in stream ciphers or one-time pads).	

Control Objectives	Test Requirements	Guidance
	7.1.c Where any algorithm or mode of operation requires a unique value per encryption operation or session, the assessor shall examine current publicly available literature or industry standards to identify security vulnerabilities in their implementation and test the software to confirm correct implementation. For example, this may include the use of a unique IV for a stream cipher mode of operation, a unique (and random) "k" value for a digital signature.	
	7.1.d Where padding is used prior to/during encryption, the assessor shall examine vendor evidence and test the software to confirm that the encryption operation always incorporates an industry-accepted standard padding method.	
	7.1.e Where hash functions are used within the software, the assessor shall: • Examine publicly available literature and research to identify vulnerable algorithms that can be exploited, and • Test the software to confirm that only approved, collision-resistant hash algorithms and methods are used with a salt value of appropriate strength, generated using a secure random number generator. ***Note:*** *The assessor should refer to Control Objective 7.3 for more information on secure random number generators.*	

Control Objectives	Test Requirements	Guidance
7.2 The software supports approved key-management processes and procedures. Approved key-management processes and procedures are those recognized by industry-standards bodies—for example: NIST, ANSI, and ISO.	**7.2.a** The assessor shall examine vendor evidence and test the software to confirm that: • All cryptographic keys that are used for providing security to critical assets—including both confidentiality and authenticity—as well as for providing other security services to the software (such as authentication of end-point or software updates) have a unique purpose. For example, no key may be used for both encryption and authentication operations. • All keys have defined generation methods, and no secret or private cryptographic keys relied upon for security of critical assets are shared between software instances, except when a common secret or private key is used for securing the storage of other cryptographic keys that are generated during the installation, initialization, or first use of the software (e.g., white-box cryptography). • All cryptographic keys have an equivalent bit strength of at least 128 bits in accordance with industry standards. • All keys have a defined crypto-period aligned with industry standards, and methods are implemented to retire and/or update each key at the end of the defined crypto-period. • The integrity and confidentiality of all secret and private cryptographic keys managed by the software are protected when stored (e.g., encrypted with a key-encrypting key that is at least as strong as the data-encrypting key and is stored separately from the data-encrypting key, or as at least two full-length key components or key shares, in accordance with an industry-accepted method). *(continued on next page)*	Whether implemented within or outside the software functionality, the manner in which cryptographic keys are managed is a critical part of the continued security of the software and the sensitive data it manages. While cryptographic key-management processes are often implemented as operational procedures, the software should support secure key-management practices based on industry standards or best practices (for example, *NIST Special Publication 800-57* or *PCI TSP Security Requirements*), including: • Generation of strong cryptographic keys • Secure cryptographic key distribution • Secure cryptographic key storage • Cryptographic key changes for keys that have reached the end of their crypto-period • Retirement or replacement of keys • Enforcement of split knowledge and dual control (when the software supports manual clear-text cryptographic key-management operations) • Prevention of unauthorized substitution of cryptographic keys • Provision of a mechanism to render irretrievable any cryptographic key material or cryptogram stored by the payment software This requirement applies to keys used to encrypt sensitive data and any respective key-encrypting keys.

Control Objectives	Test Requirements	Guidance
	7.2.a • All keys have a defined generation or injection process, and this process ensures sufficient entropy for the key. • All key-generation functions must implement one-way functions or other irreversible key-generation processes, and no reversible key calculation modes (such as key variants) are used to directly create new keys from an existing key.	
	7.2.b Where cryptography is used to protect a key, the assessor shall examine vendor evidence and test the software to confirm that security is not provided to any key by a key of lesser strength (e.g., by encrypting a 256-bit AES key with a 128-bit AES key).	
	7.2.c Where any public keys are used by the system, the assessor shall examine vendor evidence and test the software to confirm that the vendor maintains an inventory of all cryptographic keys used by the software and that the authenticity of all public keys is maintained. Vendor evidence must identify: • Key label or name • Key location • Effective and expiration date • Key purpose/type • Key length	
	7.2.d Where public or white-box keys are not unique per software instantiation the assessor shall examine vendor evidence and test the software to confirm that methods and procedures to revoke and/or replace such keys (or key pairs) exist.	

Control Objectives	Test Requirements	Guidance
	7.2.e Where the software relies upon external files or other data elements for key material (such as for public TLS certificates), the assessor shall examine vendor evidence to confirm that clear and sufficient guidance on how to install such key material in accordance with this standard—including details noting any security requirements for such key material—is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	7.2.f Where public keys are used, the assessor shall examine vendor evidence and test the software to confirm that public keys manually loaded or used as root keys are installed and stored in a way that provides dual control (to a level that is feasible on the execution environment), preventing a single user from replacing a key to facilitate a man-in-the-middle attack, easy decryption of stored data, etc. Where complete dual control is not feasible (e.g., due to a limitation of the execution environment), the assessor shall confirm that the methods implemented are appropriate to protect the public keys.	
	7.2.g The assessor shall examine vendor evidence and test the software to confirm that any secret and/or private keys are managed in a way that ensures split knowledge over the key, to a level that is feasible given the platform on which the software is executed. Where absolute split knowledge is not feasible, the assessor shall confirm that methods implemented are reasonable to protect secrets and/or private keys.	
	7.2.h The assessor shall examine vendor evidence and test the software to confirm that methods are implemented to "roll" any keys at the end of their defined crypto-period that ensure the security of the sensitive data (both cryptographic keys and data secured through use of these keys) is in line with the requirements of this standard.	

Control Objectives	Test Requirements	Guidance
7.3 All random numbers used by the software are generated using only approved random number generation (RNG) algorithms or libraries. Approved RNG algorithms or libraries are those that meet industry standards for sufficient unpredictability (e.g., *NIST Special Publication 800-22*).	**7.3.a** The assessor shall examine vendor evidence to confirm that all random number generation methods implemented in the software: • Use at least 128 bits of entropy prior to the output of any random numbers from the random number generator. • Ensure it is not possible for the system to provide or produce reduced entropy upon start-up or entry of other predictable states of the system.	Random numbers are often used with cryptography to protect sensitive information. Encryption keys and initialization values (seeds) are examples of implementations in which random numbers are required. It is not a trivial endeavor to design and implement a secure random number generator. Software vendors are required to use only approved random number generation algorithms and libraries or provide evidence to illustrate how the random number generation algorithms and libraries were tested to confirm that random numbers generated are sufficiently unpredictable. The implementation may rely on either a validated cryptographic library or module. The software vendor should have a good understanding of the installation, initialization, configuration, and usage—for example, initial seeding of the random function—of the RNG mechanisms to ensure that the implementation can meet the effective security strength required for the intended use. The calls to these libraries should also be protected from being "hooked."
	7.3.b Where the vendor is relying upon previous assessment of the random number generator, or source of initial entropy, the assessor shall examine the approval records of the previous assessment and test the software to confirm that this scheme and specific approval include the correct areas of the software in the scope of its assessment, and that the vendor claims do not exceed the scope of the evaluation or approval of that software. For example, some cryptographic implementations approved under FIPS 140-2 or 140-3 require seeding from an external entropy source to correctly output random data.	
	7.3.c Where third-party software, platforms, or libraries are used for all or part of the random number generation process, the assessor shall examine current publicly available literature to confirm that there are no publicly known vulnerabilities or concerns with the software that may compromise its use for generating random values in the software under test. Where problems are known, but have been mitigated by the software vendor, the assessor shall examine vendor evidence and test the software to confirm that the vulnerabilities have been sufficiently mitigated. The assessor shall test the software to confirm that third-party software, platforms, or libraries are correctly integrated, implemented, and configured.	

Control Objectives	Test Requirements	Guidance
	7.3.d The assessor shall examine vendor evidence and test the software to confirm that methods have been implemented to prevent or detect (and respond) the interception, or "hooking," of random number calls that are serviced from third-party software, or the platform on which the software is executed.	
	7.3.e The assessor shall test the software to obtain 128MB of data output from each random number generator implemented in the system to confirm the lack of statistical correlation in the output. This data may be generated by the assessor directly, or supplied by the vendor, but the assessor must confirm that the generation method implemented ensures that the data is produced as it would be produced by the software during normal operation. ***Note:*** *The assessor can use the NIST Statistical Test Suite to identify statistical correlation in the random number generation implementation.*	
7.4 Random values have entropy that meets the minimum effective strength requirements of the cryptographic primitives and keys that rely on them.	**7.4.a** The assessor shall examine vendor evidence and test the software to confirm that the methods used for the generation of all cryptographic keys and other material (such as IVs, "k" values for digital signatures, etc.) have entropy that meets the minimum effective strength requirements of the cryptographic primitives and keys.	Entropy is the degree of randomness of a random value generator. The higher the entropy, the less predictable the next value in a random number generator is likely to be. Note that a non-deterministic random number generator (NDRG) may produce an output string that contains less entropy than implied by the length of the output. A deterministic random number generator (DRNG) is dependent on the entropy of its seed value.

Control Objectives	Test Requirements	Guidance
	7.4.b Where cryptographic keys are generated through processes which require direct user interaction, such as through the entry of a passphrase or the use of "random" user interaction with the software, the assessor shall examine vendor evidence and test the software to confirm that these processes are implemented in such a way that they provide sufficient entropy. Specifically, the assessor shall confirm that: • Any methods used for generating keys directly from a password/passphrase enforces an input domain that is able to provide sufficient entropy, such that the total possible inputs are at least equal to that of the equivalent bit strength of the key being generated (e.g., a 32-hex-digit input field for an AES128 key). • The passphrase is passed through an industry-standard key-derivation function, such as PBKDF2 or bcrypt, which extends the work factor for any attempt to brute-force the passphrase value. The assessor shall confirm that a work factor of at least 10,000 is applied to any such implementation. • Clear and sufficient guidance is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1 that any passphrase used must be: – Randomly generated itself, using a valid and secure random process: an online random number generator must not be used for this purpose. – Never implemented by a single person, such that one person has an advantage in recovering the clear key value; violating the requirements for split knowledge.	

Control Objectives	Test Requirements	Guidance
	7.4.c Where any third-party software or platforms are relied upon by the software and do not meet the entropy requirements, the assessor shall examine vendor evidence and test the software to confirm that sufficient mitigations are implemented, and that clear and sufficient guidance on the secure configuration and usage of these software components is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	

Secure Software Operations

The software vendor facilitates secure software operation.

Control Objectives	Test Requirements	Guidance
Control Objective 8: Activity Tracking All software activity involving critical assets is tracked.		
8.1 All access attempts and usage of critical assets is tracked and traceable to a unique individual. ***Note:*** *This Secure Software Standard recognizes that some execution environments cannot support the detailed logging requirements in other PCI standards. Therefore, the term "activity tracking" is used here to differentiate the expectations of this standard with regards to logging from similar requirements in other PCI standards.*	**8.1** The assessor shall examine vendor evidence and test the software to confirm that all access attempts and usage of critical assets are tracked and traceable to a unique identification for the person, system, or entity performing the access.	To facilitate user accountability and to allow post-incident forensic investigation, payment software should capture and maintain historical records of all software activity involving critical assets (i.e., sensitive data and resources and usage of sensitive functions), and ensure such activity is traceable to a unique user (e.g., person), system, or other entity accessing critical assets. Examples of activities that the software should record include: • All individual user attempts to access sensitive data or resources • Usage of or changes to sensitive functions, such as the software's identification and authentication mechanisms or activity tracking mechanisms • Initialization, stopping, or pausing of sensitive functions This Control Objective does not mandate the logging of each encryption operation or function processing sensitive data, but it does require that access is tracked and any methods that may expose sensitive data are also tracked.

Control Objectives	Test Requirements	Guidance
8.2 All activity is captured in sufficient and necessary detail to accurately describe what specific activities were performed, who performed them, the time they were performed, and which critical assets were impacted.	**8.2.a** The assessor shall examine vendor evidence and test the software to confirm that the tracking method(s) implemented capture specific activity performed, including: • Enablement of any privileged modes of operation • Disabling of encryption of sensitive data • Decryption of sensitive data • Exporting of sensitive data to other systems or processes • Failed authentication attempts • Disabling or deleting a security control or altering security functionality	By recording the details in this requirement for all attempts to access or use critical assets, malicious activity or potential software or data compromise can be quickly identified and with sufficient detail to know who performed the activity, whether the attempt was successful, when the activity occurred, what critical assets were affected, and the origination of the event.
	8.2.b The assessor shall examine vendor evidence and test the software to confirm that the tracking method(s) implemented provide: • A unique identification for the person, system, or entity performing the access • A timestamp for each tracked event • Details on what critical asset has been accessed	
	8.2.c The assessor shall test the software to confirm that sensitive data is not directly recorded in the tracking data.	
8.3 The software supports secure retention of detailed activity records.	**8.3.a** Where the activity records are managed by the software, including only temporarily before being passed to other systems, the assessor shall examine vendor evidence and test the software to confirm that the protection methods are implemented to protect completeness, accuracy, and integrity of the activity records.	In order to identify anomalous behavior and to facilitate forensic investigation upon suspicion of potential software or data compromise, the software must facilitate the retention of detailed activity records either through native functionality (i.e., within the software itself) or support integration with other solutions such as centralized log servers, cloud-based logging solutions, or a back-end monitoring solution. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	8.3.b Where the software utilizes other systems for maintenance of tracking data, such as a log server, the assessor shall examine vendor evidence to confirm that clear and sufficient guidance on the correct and complete setup and/or integration of the software with the log storage system is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1. The assessor shall test the software to confirm methods are implemented to secure the authenticity of the tracking data during transmission to the log storage system, and to confirm that this protection meets the requirements of this standard—for example, authenticity parameters must be applied using strong cryptography—and any account or authentication parameters used for access to an external logging system are protected.	Without adequate protection of activity records, their completeness, accuracy, and integrity cannot be guaranteed, and any reliance that would otherwise be placed on them (such as during a forensic investigation) would be negated. When activity records are managed by the software, the records must be protected in accordance with all applicable requirements for the protection of sensitive data.
8.4 The software handles failures in activity-tracking mechanisms such that the integrity of existing activity records is preserved.	**8.4.a** The assessor shall examine vendor evidence and test the software to confirm that failure of the activity tracking system does not violate the integrity of existing records. The assessor shall explicitly confirm that: • The software does not overwrite existing tracking data upon a restart of the software. Each new start shall only append to existing datasets or shall create a new tracking dataset. • Where unique dataset names are relied upon for maintaining integrity between execution instances, the implementation ensures that other software (including another instance of the same software) cannot overwrite or render invalid existing datasets. • Where possible the software applies suitable file privileges to assist with maintaining the integrity of the tracking dataset (such as applying an append only access control to a dataset once created). Where the software does not apply such controls, the assessor shall confirm reasonable justification exists describing why this is the case, why the behavior is sufficient, and what additional mitigations are applied to maintain the integrity of the tracking data.	Software security controls should be implemented to ensure that when activity-tracking mechanism(s) fail, those failures are handled in a way that maintains the integrity and confidentiality (if applicable) of the records.

Control Objectives	Test Requirements	Guidance
	8.4.b The assessor shall examine vendor evidence, including source code, and shall test the software, including (wherever possible): • Performing actions that should be tracked, force-closing and then restarting the software, and performing other tracked actions. • Performing actions that should be tracked, power-cycling the platform on which the software is executing, and then restarting the software and performing other tracked actions. • Locking access to the tracking dataset and confirming that the software handles the inability to access this dataset in a secure way, such as by creating a new dataset or preventing further use of the software. • Preventing the creation of new dataset entries by preventing further writing to the media on which the dataset is located (e.g., by using media that has insufficient available space). Where any of the tests above are not possible, the assessor shall interview personnel to confirm reasonable justification exists to describe why this is the case and shall confirm protections are put in place to prevent such scenarios from affecting the integrity of the tracking records.	

Control Objectives	Test Requirements	Guidance
Control Objective 9: Attack Detection Attacks are detected, and the impacts/effects of attacks are minimized.		
9.1 The software detects and alerts upon detection of anomalous behavior, such as changes in post-deployment configurations or obvious attack behavior.	**9.1.a** The assessor shall examine vendor evidence and test the software to confirm that, where possible, the software implements a method to validate the integrity of its own executable and any configuration options, files, and datasets that it relies upon for operation (such that unauthorized, post-deployment changes can be detected). Where the execution environment prevents this, the assessor shall examine vendor evidence and current publicly available literature on the platform and associated technologies to confirm that there are indeed no methods for validating authenticity. The assessor shall then test the software to confirm controls are implemented to minimize the associated risk.	Software should possess basic functionality to differentiate between normal and anomalous user behavior. Examples of anomalous behavior that should be automatically detected by the software include changes in post-deployment (or post-initialization) configurations or obvious automated-attack behaviors—e.g., frequent input of a password can be an indicator of brute-force attempts.
	9.1.b The assessor shall examine vendor evidence and test the software to confirm that integrity values used by the software and dataset(s) upon which it relies for secure operation are checked upon software execution, and at least every 36 hours thereafter (if the software continues execution during that time period). The assessor shall confirm what action the software takes upon failure of these checks and confirm that the processing of sensitive data is halted until this problem is remediated.	
	9.1.c Where cryptographic primitives are used by any anomaly-detection methods, the assessor shall examine vendor evidence and test the software to confirm that cryptographic primitives are protected.	
	9.1.d Where stored values are used by any anomaly-detection methods, the assessor shall examine vendor evidence and test the software to confirm that these values are considered sensitive data and protected accordingly.	

Control Objectives	Test Requirements	Guidance
	9.1.e Where configuration or other dataset values can be modified by the software during execution, the assessor shall examine vendor evidence and test the software to confirm that integrity protections are implemented to allow for this update while still ensuring dataset integrity can be validated after the update.	
	9.1.f The assessor shall examine vendor evidence and test the software to confirm that the software implements controls to prevent brute-force attacks on account, password, or cryptographic-key input fields (e.g., input rate limiting).	

Secure Software Lifecycle Management

The software vendor implements secure software lifecycle management practices.

Control Objectives	Test Requirements	Guidance
Control Objective 10: Threat and Vulnerability Management The software vendor identifies, assesses, and manages threats and vulnerabilities to its payment software.		
10.1 Software threats and vulnerabilities are identified, assessed, and addressed.	**10.1.a** The assessor shall examine vendor evidence to confirm that the vendor identifies common methods for attack against the software product. This may include platform-level, protocol-level, and/or language-level attacks.	Determining how to effectively secure and defend the software against attacks requires a thorough understanding of the specific threats and potential vulnerabilities applicable to the vendor's software. This typically involves understanding the following: • The types of information collected, stored, processed, or transmitted by the software; • The motivations an attacker may have for attacking the software; • The methods an attacker might utilize or the vulnerabilities an attacker might try to exploit during an attack; • The exploitability of any identified vulnerabilities; and • The impact a successful attack. The identified threats and vulnerabilities should be tracked, assigned to responsible personnel, and fixed or mitigated prior to payment software release. For guidance on threat analysis and cyber-resiliency design principles, refer to industry standards and guidance such as *NIST Special Publication 800-160,* Appendix F.
	10.1.b The assessor shall examine vendor evidence to confirm that the list of common attacks is valid for the software the vendor has produced and shall note where this does not include common attack methods detailed in industry-standard references such as OWASP and CWE lists.	
	10.1.c The assessor shall examine vendor evidence to confirm that mitigations against each identified attack vector exists, and that the vendor's software release process includes validation of the existence of these mitigations.	

Control Objectives	Test Requirements	Guidance
10.2 Vulnerabilities in the software and third-party components are tested for and fixed prior to release.	**10.2.a** The assessor shall examine vendor evidence to confirm that the software vendor has implemented robust testing processes throughout the software lifecycle to validate the mitigations used to secure the software against attacks outlined in the vendor threat model and vulnerability assessment.	Most software vulnerabilities are introduced as a result of coding errors, bad design, improper implementation of software functionality, or the use of vulnerable components. Software should be developed and tested in a manner that minimizes the existence of any vulnerabilities and detects those that emerge over time, such that the vulnerabilities may be addressed before the software is released or updated. Techniques to avoid the introduction of vulnerabilities during development include the use of security coding practices, testing code during each phase of the development lifecycle using automated tools (such as static/dynamic analysis tools, interactive security testing tools, etc.), and standardizing the use of known secure components (e.g., common code that has already undergone significant security vetting). To minimize the introduction of software vulnerabilities from third-party components, those components must also be evaluated. Ideally, they should be subject to the same secure development and testing processes as the software created by the vendor. 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.

Control Objectives	Test Requirements	Guidance
	10.2.b The assessor shall examine evidence, including documented testing processes and output of several instances of the testing, as performed on the software under evaluation to confirm that the testing process: • Includes, at a minimum, the use of automated tools capable of detecting vulnerabilities both in software code and during software execution. • Includes the use of tools for security testing that are appropriate for detecting applicable vulnerabilities and are suitable for the software architecture, development languages, and frameworks used in the development of the software. • Accounts for the entire code base, including detecting vulnerabilities in third-party, open-source, or shared components and libraries. • Accounts for common vulnerabilities and attack methods. • Demonstrates a history of finding software vulnerabilities and remediating them prior to retesting of the software.	
	10.2.c Where vendor evidence shows the release of software with known vulnerabilities, the assessor shall examine vendor evidence to confirm that: • The vendor implements an industry-standard vulnerability-ranking system (such as CVSS) that allows for the categorization of vulnerabilities. • For all vulnerabilities, the vendor provides a remediation plan—it is unacceptable for a known vulnerability to remain unmitigated for an indefinite period.	

Control Objectives	Test Requirements	Guidance
Control Objective 11: Secure Software Updates The software vendor facilitates secure software releases and updates.		
11.1 Software updates to fix known vulnerabilities are made available to stakeholders in a timely manner.	**11.1.a** The assessor shall examine vendor evidence to confirm that: • Reasonable criteria exist for releasing software updates to fix security vulnerabilities. • Security updates are made available to stakeholders in accordance with defined criteria.	Vulnerabilities in software should be fixed as soon as possible to enable software users and other stakeholders to address any risks before vulnerabilities in their payment systems and software can be exploited by attackers. Vulnerabilities should be addressed in a manner that is commensurate with the risk they pose to software users or other stakeholders. The most critical or severe vulnerabilities (i.e., those with the highest exploitability and the greatest potential impact to stakeholders) should be patched immediately, followed by those with moderate-to-low exploitability or potential impact. The criteria for determining how and when to make patches available to stakeholders should be defined and followed.
	11.1.b For a sample of vendor software updates, the assessor shall examine vendor evidence, including update-specific security-testing results and details, to confirm security fixes have been made available to stakeholders in accordance with defined criteria. Where updates were not provided in accordance with defined criteria, such instances are be reasonably justified by the vendor.	
11.2 Software releases and updates are delivered in a secure manner that ensures the integrity of the software and its code.	**11.2.a** The assessor shall examine vendor evidence to confirm that the method by which the vendor releases software updates ensures the integrity of the software and its code during transmission and install. Where user instructions are required to validate the integrity of the code, the assessor shall confirm that clear and sufficient guidance to enable the process to be correctly performed is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	Security updates should include a mechanism within the update process to verify the update code has not been replaced or tampered with. Examples of integrity checks include, but are not limited to, checksums and digitally signed certificates (where implemented appropriately), etc. Verification could be implemented within the software itself, or instruction (e.g., release notes) can be provided by the vendor to allow verification of the software updates by its customers. In addition, the process of distributing updates and patches should prevent malicious individuals from intercepting the updates in transit, modifying them, and then redistributing them to unsuspecting customers.
	11.2.b Where the integrity method implemented is not cryptographically secure (such as through the use of digital signatures), the assessor shall examine vendor evidence to confirm that the software distribution method provides a chain of trust (such as through use of a TLS connection that provides compliant cipher‑suite implementations).	

Control Objectives	Test Requirements	Guidance
	11.2.c The assessor shall examine vendor evidence to confirm that the software vendor informs users of the software updates, and that clear and sufficient guidance on how they may be obtained and installed is provided in the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1.	
	11.2.d The assessor shall examine vendor evidence to confirm the vendor has a process for informing users of the software of known vulnerabilities that have not yet been patched by a new version of the software. This includes vulnerabilities that may exist in third-party software and libraries used by the vendor's software product. The assessor shall confirm that this process includes providing the users with suggested mitigations for any such vulnerabilities.	
	11.2.e The assessor shall examine vendor evidence to confirm the update mechanisms cover all software, configuration files, and other metadata that may be used by the software for security purposes, or which may in some way affect security.	

Control Objectives	Test Requirements	Guidance
Control Objective 12: Software Vendor Implementation Guidance The software vendor provides stakeholders with clear and thorough guidance on the secure implementation, configuration, and operation of the software.		
12.1 The software vendor provides stakeholders with clear and thorough guidance on the secure implementation, configuration, and operation of its payment software.	**12.1.a** The assessor shall examine vendor evidence to confirm that the vendor creates and provides, to all stakeholders, clear and sufficient guidance to allow for the secure installation and use of the software.	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 software accounts, services, and features • Changes in resource access permissions • Integration with third-party cryptographic libraries, random number generators, etc. The provided guidance should result in a secure configuration across all configurable options.
	12.1.b The assessor shall examine vendor evidence to confirm that the guidance: • Includes details on how to securely and correctly install any third-party software that is required for the operation of the vendor software. • Provides instructions on the correct configuration of the platform(s) on which the software is to be executed, including setting security parameters and installation of any data elements (such as certificates). • Includes instructions for key management (e.g., use of keys, how keys are distributed, loaded, removed, changed, destroyed, etc.) • Does not instruct the user to disable security settings or parameters within the installed environment, such as anti-malware software or firewall or other network-level protection systems. • Does not instruct the user to execute the software in a privileged mode higher than what is required by the software. *(continued on next page)*	

Control Objectives	Test Requirements	Guidance
	12.1.b • Provides details on how to validate the version of the software and clearly indicates for which version(s) of the software the guidance is written. • Provides justification for any requirements in this standard that are to be assessed as not applicable. For each of these, the assessor shall confirm reasonable justification exists for why this is the case and confirm that it agrees with their understanding and the results of their testing of the software.	

Module A – Account Data Protection Requirements

Module Name	Overview	Control Objectives
Module A – Account Data Protection Requirements	Security requirements for software that stores, processes, or transmits account data.	**A.1**: Sensitive Authentication Data **A.2**: Cardholder Data Protection

Purpose and Scope

This section (hereinafter referred to as the "Account Data Protection Module") defines security requirements and assessment procedures for software that stores, processes, or transmits account data. For the purposes of this module, account data is defined as follows:

Account Data	
Cardholder Data includes:	**Sensitive Authentication Data includes:**
▪ Primary Account Number (PAN) ▪ Cardholder Name ▪ Expiration Date ▪ Service Code	▪ Full track data (magnetic-stripe data or equivalent on a chip) ▪ CAV2/CVC2/CVV2/CID ▪ PINs/PIN blocks

The primary account number (PAN) is the defining factor for cardholder data. If PAN is stored, processed, or transmitted or is otherwise present, the requirements in this module apply in addition to the Secure Software Core Requirements.

The table on the following page illustrates commonly used elements of cardholder data and sensitive authentication data, whether storage of that data is permitted or prohibited, and whether this data needs to be protected. This table is not meant to be exhaustive, but it is presented to illustrate the different types of requirements that apply to each data element.

		Data Element	Storage Permitted	Render Stored Data Unreadable per Control Objective A.2.3
Account Data	**Cardholder Data**	Primary Account Number (PAN)	Yes	Yes
		Cardholder Name	Yes	No
		Service Code	Yes	No
		Expiration Date	Yes	No
	Sensitive Authentication Data[3]	Full Track Data [4]	No	Cannot store per Control Objective A.1.1
		CAV2/CVC2/CVV2/CID[5]	No	Cannot store per Control Objective A.1.1
		PIN/PIN Block[6]	No	Cannot store per Control Objective A.1.1

Control Objectives A.2.2 and A.2.3 apply only to PAN. If PAN is stored with other elements of cardholder data, only the PAN must be rendered unreadable according to Control Objective A.2.3. Sensitive authentication data must not be stored after authorization, even if encrypted, unless the software is intended only for use by issuers or organizations that support issuing services. Only in those cases can sensitive authentication data be stored post-authorization.

3 *Sensitive authentication data must not be stored after authorization (even if encrypted).*

4 *Full track data from the magnetic stripe, equivalent data on the chip, or elsewhere.*

5 *The three- or four-digit value printed on the front or back of a payment card.*

6 *Personal identification number entered by cardholder during a card-present transaction, and/or encrypted PIN block present within the transaction message.*

Account Data Protection

The confidentiality of Account Data is protected.

Control Objectives	Test Requirements	Guidance
Control Objective A.1: Sensitive Authentication Data Sensitive authentication data is not retained after authorization.		
A.1.1 The software does not store sensitive authentication data after authorization—even if encrypted—unless the software is intended only for use by issuers or organizations that support issuing services.	**A.1.1.a** For each instance of sensitive authentication data identified in Control Objective 1.1, the assessor shall test the software, including generation of error conditions and log entries, and usage of forensic tools and/or methods, to identify all potential storage locations and to confirm that the software does not store sensitive authentication data after authorization. This includes temporary storage (such as volatile memory), semi-permanent storage (such as RAM disks), and non-volatile storage (such as magnetic and flash storage media). **A.1.1.b** Where sensitive authentication data is stored after authorization, the assessor shall examine vendor evidence to confirm the software is intended only for use by issuers or organizations that support issuing services.	Sensitive authentication data consists of full track data, card validation code or value, and PIN data. Storage of sensitive authentication data after authorization is prohibited. This data is very valuable to malicious individuals as it allows them to generate counterfeit payment cards and create fraudulent transactions. Testing should include at least the following types of files, as well as any other output generated by the payment software: • Incoming transaction data • All logs (for example, transaction, history, debugging, error) • History files • Trace files • Audio and image files (digital voice, biometrics, etc.) • Non-volatile memory, including non-volatile cache • Database schemas • Database contents

Control Objectives	Test Requirements	Guidance
Control Objective A.2: Cardholder Data Protection Stored cardholder data is protected.		
A.2.1 The software vendor provides guidance to customers regarding secure deletion of cardholder data after expiration of the customer-defined retention period.	**A.2.1** The assessor shall examine the instructions prepared by the software vendor and confirm the documentation includes the following guidance for customers, integrators, and resellers: • A list of all locations where the software stores cardholder data. • Instructions on how to securely delete cardholder data stored by the payment software, including data stored on underlying software or systems (such as OS, databases, etc.). • Instructions for configuring the underlying software or systems (such as OS, databases, etc.) to prevent inadvertent capture or retention of cardholder data—for example, system backup or restore points.	The software vendor must provide details of all locations where the software may store cardholder data, including in any underlying software or systems (such as OS, databases, etc.), as well as instructions for securely deleting the data from these locations once the data has exceeded the customer's defined retention period. Customers and integrators/resellers must also be provided with configuration details for the underlying systems that the software runs on, to ensure these underlying systems are not capturing cardholder data without the customer's knowledge. The customer needs to know how the underlying systems could be capturing data from the software so they can either prevent it from being captured or ensure the data is properly protected.
A.2.2 The software masks the PAN such that only a maximum of the first six and last four digits are displayed by default.	**A.2.2.a** The assessor shall examine vendor evidence, including the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1, to confirm the guidance includes the following instructions for customers and integrators/resellers: • Details of all instances where PAN is displayed. • Confirmation that the payment software masks PAN to display a maximum of the first six and last four digits by default on all displays. • Instructions for how to configure the software to display more than the first six/last four digits of the PAN (includes displays of the full PAN).	The display of full PAN on items such as computer screens, payment card receipts, logs, faxes, or paper reports can result in this data being obtained by unauthorized individuals and used fraudulently. The masking approach should always ensure that only the minimum number of digits is displayed as necessary to perform a specific business function. For example, if only the last four digits are needed to perform a business function, mask the PAN so that individuals performing that function can view only the last four digits.

Control Objectives	Test Requirements	Guidance
	A.2.2.b The assessor shall test the software to confirm that all displays of PAN are masked by default.	
	A.2.2.c The assessor shall examine vendor evidence and test the software to confirm that for each instance where the PAN is displayed, the instructions for displaying more than the first six/last four digits are accurate.	
A.2.3 Render the PAN unreadable anywhere it is stored (including data on portable digital media, backup media, and in logs) by using any of the following approaches: • Truncation (hashing cannot be used to replace the truncated segment of PAN). • Index tokens and pads (pads must be securely stored). • Strong cryptography with associated key-management processes and procedures.	**A.2.3.a** The assessor shall examine vendor evidence, including the software vendor's implementation guidance made available to stakeholders per Control Objective 12.1 to verify the guidance includes the following: • Details of any configurable options for each method used by the software to render cardholder data unreadable, and instructions on how to configure each method for all locations where cardholder data is stored by the payment software. • A list of all instances where cardholder data may be output for the customer to store outside of the payment application, and instructions that the customer is responsible for rendering the PAN unreadable in all such instances. • Instruction that if debugging logs are ever enabled (for example, for troubleshooting purposes) and they include the PAN, they must be protected, disabled as soon as troubleshooting is complete, and securely deleted when no longer needed.	Lack of protection of PANs can allow malicious individuals to view or download this data. The intent of truncation is that only a portion (not to exceed the first six and last four digits) of the PAN is stored. An index token is a cryptographic token that replaces the PAN based on a given index for an unpredictable value. A one-time pad is a system in which a randomly generated secret key is used only once to encrypt a message that is then decrypted using a matching one-time pad and key. The intent of strong cryptography is that the encryption be based on an industry-tested and accepted algorithm (not a proprietary or "home-grown" algorithm), with strong cryptographic keys.

Control Objectives	Test Requirements	Guidance
	A.2.3.b The assessor shall test the software to confirm that the method used to protect the PAN, including the encryption algorithms (if applicable), and verify that the PAN is rendered unreadable using any of the following methods: • Truncation • Index tokens and pads, with the pads being securely stored • Strong cryptography, with associated key-management processes and procedures. ***Note:*** *The assessor should examine several tables, files, log files and any other resources created or generated by the software to verify the PAN is rendered unreadable.*	
	A.2.3.c Where software creates both tokenized and truncated versions of the same PAN, the assessor shall test the software to confirm that the tokenized and truncated versions cannot be correlated to reconstruct the original PAN.	
	A.2.3.d Where software creates or generates files for use outside the software—for example, files generated for export or backup—including for storage on removable media, the assessor shall test the software, including examining a sample of generated files, such as those generated on removable media (for example, back-up tapes), to confirm that the PAN is rendered unreadable.	
	A.2.3.e If the software vendor stores the PAN for any reason (for example, because log files, debugging files, and other data sources are received from customers for debugging or troubleshooting purposes), the assessor shall examine vendor evidence and test the software to confirm that the PAN is rendered unreadable in accordance with this requirement.	

Module B – Terminal Software Requirements

Module Name	Overview	Control Objectives
Module B: Terminal Software Requirements	Security requirements for software intended for deployment and execution on PCI-approved POI devices.	**B.1**: Terminal Software Documentation **B.2**: Terminal Software Design **B.3**: Terminal Software Attack Mitigation **B.4**: Terminal Software Security Testing **B.5**: Terminal Software Implementation Guidance

Purpose and Scope

This section (hereinafter referred to as the "Terminal Software Module" or "this module") defines security requirements and assessment procedures for software intended for deployment and execution on PCI-approved POI devices. Terminal software is developed explicitly for deployment and execution on PCI-approved POI devices and does not meet the definition of Firmware as defined in the *PCI PIN Transaction Security (PTS) Point-of-Interaction (POI) Modular Security Requirements* (hereinafter referred to as the "PCI PTS POI Standard").

Background

PCI-approved POI devices provide a high degree of confidentiality and integrity protection for payment data and payment transactions through the implementation of strict physical and logical protection mechanisms. Software that is deployed and executed on PCI-approved POI devices must not degrade or adversely affect the protection mechanisms provided by the device. In addition, the software must not provide features or functions that could facilitate or allow those protection mechanisms to be circumvented or rendered ineffective.

The requirements and assessment procedures defined in the Terminal Software Module have been developed to help ensure that terminal software protects payment data and does not introduce features, functions, or weaknesses that could enable an attacker to circumvent or render ineffective the protection mechanisms provided by the underlying PCI-approved POI devices upon which the software is intended to be deployed.

Terminal Software Evaluation

In addition to the requirements and assessment procedures defined in the Terminal Software Module, the requirements and assessment procedures defined in the Secure Software Core Requirements section of this document (i.e., the "Core Module") also apply to terminal software. Some of the requirements defined within the Terminal Software Module are extensions of Core Module requirements. Where such extensions are noted, the Terminal Software Requirements should be evaluated in conjunction with their associated Secure Software Core Requirements.

Where the terminal software stores, processes, or transmits cardholder data (CHD) and/or sensitive authentication data (SAD), the requirements and assessment procedures defined within Module A: Account Data Protection Requirements (i.e., the "Account Data Protection Module") shall also apply.

Additional Considerations

Some assessment procedures in this module require examination of documentation describing the security features and functions of the underlying payment terminal. The terminal software vendor should work with their assessor(s)—as well as the respective payment terminal vendors for each of the devices to be included as part of the terminal software evaluation—to identify and compile all device documentation needed for the terminal software evaluation. For more information about Secure Software assessment preparation and activities, please refer to the *Secure Software Program Guide*.

Terminal Software Security

Terminal software protects account data from unauthorized disclosure and ensures the underlying payment terminal's security features, functions and characteristics are not circumvented or otherwise rendered ineffective.

Control Objectives	Test Requirements	Guidance
Control Objective B.1: Terminal Software Documentation The software architecture is documented and includes diagrams that describe all software components and services in use and how they interact.		
B.1.1 The software vendor maintains documentation that describes all software components, interfaces, and services provided or used by the software.	**B.1.1** The assessor shall examine all relevant documentation and evidence necessary to confirm that the software vendor maintains documentation describing the software's overall design and function including, but not limited to, the following: • All third-party and open-source components, external services, and Application Programming Interfaces (APIs) used by the software. • All User Interfaces (UI) and APIs provided or made accessible by the software.	Software vendors should also maintain detailed documentation that clearly and effectively describes the overall design and function of its software, including all services (internal and external), components, and functions used and provided by the software, and how those services, components, and functions interact.
B.1.2 The software vendor maintains documentation that describes all data flows and functions that involve sensitive data. ***Note:*** *This control objective is an extension of Control Objectives 1.1 and 1.2. Validation of these control objectives should be performed at the same time.*	**B.1.2.a** The assessor shall examine all relevant documentation and evidence necessary to confirm that the software vendor maintains documentation describing all sensitive data flows including, but not limited to, the following: • All sensitive data stored, processed, or transmitted by the software. • All locations where sensitive data is stored, including both temporary and persistent storage locations. • How sensitive data is securely deleted from storage (both temporary and persistent) when no longer needed.	In addition to identifying the components, interfaces, and services exposed by the software, the software vendor should also maintain documentation that clearly identifies and describes the types of data stored, processed, and transmitted by the software; whether and how that data is shared between components and functions; and the protection mechanisms implemented or relied upon by the software to protect that data. This type of documentation clarifies how data is stored, processed, or transmitted by the software, with whom the data is shared, and how the software may be attacked to gain access to the software's critical assets.

Control Objectives	Test Requirements	Guidance
	B.1.2.b The assessor shall examine all relevant documentation and evidence necessary to confirm that the software vendor maintains documentation describing all functions that handle sensitive data including, but not limited to, the following: • All inputs, outputs, and possible error conditions for each function that handles sensitive data. • All cryptographic algorithms, modes of operation, and associated key management practices for all functions that employ cryptography for the protection of sensitive data.	
B.1.3 The software vendor maintains documentation that describes all configurable options that can affect the security of sensitive data.	**B.1.3** The assessor shall examine all relevant documentation and evidence necessary to confirm that the software vendor maintains documentation describing all configurable options provided or made available by the software that can impact the security of sensitive data including, but not limited to, the following: • All configurable options that could allow access to sensitive data. • All configurable options that could enable modification of any mechanisms used to protect sensitive data. • All remote access features, functions, and parameters provided or made available by the software. • All remote update features, functions, and parameters provided or made available by the software. • The default settings for each configurable option.	Software vendors should identify all configurable options available within their software, particularly those that control security features and functions. Configurable features must be considered as potential avenues for attacking the software. Where configurable options enable control over security features and functions, robust security controls should be implemented to protect the configurable security features from being misused. Additionally, all configurable options should be configured to their most secure settings by default in accordance with Control Objective 2.

Control Objectives	Test Requirements	Guidance
Control Objective B.2: Terminal Software Design The software does not implement any feature that enables the security features, functions, and characteristics of the payment terminal to be circumvented or rendered ineffective.		
B.2.1 The software is intended for deployment and operation on payment terminals (i.e., PCI-approved POI devices).	**B.2.1** The assessor shall examine all relevant software documentation and evidence necessary to determine the payment terminals upon which the software is to be deployed. For each of the payment terminals identified in the software documentation and included in the software assessment, the assessor shall examine the payment terminal's device characteristics and compare them with the following characteristics specified in the *PCI SSC's List of Approved PTS Devices* to confirm they match: • Model name/number • PTS approval number • Hardware version number • Firmware version number(s)	Payment terminals provide a high degree of confidentiality and integrity protection for payment data and payment transactions through the implementation of strict physical and logical protection mechanisms. Software that is deployed and executed on these payment terminals should use the approved features and functions provided by the payment terminal rather than implementing its own equivalent features or functions, to avoid exposing vulnerabilities or other weaknesses that could allow an attacker to circumvent or render ineffective the security features of the payment terminal.
B.2.2 The software uses only the external communication methods included in the payment terminal's PTS device evaluation. ***Note:*** *The payment terminal may provide an IP stack approved per the PTS Open Protocols module, or the device may provide serial ports or modems approved by the PTS evaluation to communicate transaction data encrypted by its PCI PTS SRED functions. Using any external communication methods not included in the PCI-approved POI device evaluation invalidates the PTS approval, and such use is prohibited for terminal software.*	**B.2.2.a** The assessor shall examine all relevant software documentation and source code necessary to determine whether the software supports external communications. **B.2.2.b** Where the software supports external communications, the assessor shall examine all relevant payment terminal documentation—including the payment terminal vendor's security guidance/policy—to determine which external communication methods were included in the payment terminal's PTS device evaluation. **B.2.2.c** The assessor shall examine all relevant software documentation and source code necessary to confirm that the software uses only the external communication methods included in the payment terminal's PTS device evaluation and does not implement its own external communication methods (i.e., its own IP stack).	To ensure software does not degrade or defeat the security mechanisms provided by the underlying payment terminal, the software must use the device-provided security features and functions in accordance with the payment terminal vendor's security guidance/policy. This is particularly true for external communication methods. Under no circumstances should the software provide its own communication methods (e.g., VMs, IP stack, scripting languages, etc.) to control device-level interfaces. The introduction of any such function by the software could introduce new vulnerabilities or weaknesses that would allow malicious entities to circumvent the security protections provided by the payment terminal and degrade the overall security characteristics of both the software and the underlying device.

Control Objectives	Test Requirements	Guidance
B.2.2.1 Where the software relies on the Open Protocols feature of the payment terminal, the software is developed in accordance with the payment terminal vendor's security guidance/policy.	**B.2.2.1** The assessor shall examine all relevant payment terminal documentation—including the payment terminal vendor's security guidance/policy—and all relevant software vendor process documentation and software design documentation to confirm that the software is developed in accordance with the payment terminal vendor's security guidance/policy.	Payment terminal vendor security guidance/policy is intended for application developers, system integrators, and end-users of the platform to meet the PCI PTS POI Open Protocol (as well as other PTS) requirements as part of a PCI-approved POI device evaluation.
B.2.2.2 The software does not circumvent, bypass, or add additional services or protocols to the Open Protocols of the payment terminal as approved and documented in the payment terminal vendor's security guidance/policy. This includes the use of: • Link Layer protocols • IP protocols • Security protocols • IP Services	**B.2.2.2** The assessor shall examine all relevant software documentation and source code to confirm that the software does not circumvent, bypass, or add additional services or protocols to the Open Protocols of the payment terminal as approved and documented in the payment terminal vendor's security guidance/policy. This includes the use of: • Link Layer protocols • IP protocols • Security protocols • IP Services	The Open Protocol requirements in the *PCI PTS POI Standard* ensure that open protocols and services in payment terminals do not have vulnerabilities that can be remotely exploited and yield access to sensitive data or sensitive resources in the payment terminal. The payment terminal vendor defines what protocols and services are supported by the payment terminal and provides guidance to their use. Adding or enabling additional services or protocols or failing to follow the issued payment terminal vendor's security guidance/policy invalidates the approval status of that device for that implementation.
B.2.3 The software does not bypass or render ineffective any encryption methods or account data security methods implemented by the payment terminal in accordance with the payment terminal vendor's security guidance/policy.	**B.2.3.a** The assessor shall examine all relevant software documentation and source code necessary to determine whether the software facilitates encryption of sensitive data. Where the software does provide such a function, the assessor shall confirm the software does not bypass or render ineffective any encryption methods or account data security methods implemented by the payment terminal as follows:	Payment terminals are designed to provide robust cryptographic and key management functions. For example, PCI PTS POI-approved devices are verified to meet stringent requirements for loading, managing, and protecting cryptographic keys. Software that provides its own data encryption methods must not include methods that would enable an attacker to bypass or render ineffective the encryption methods implemented by the payment terminal and required by the payment terminal vendor's security guidance/policy.

Control Objectives	Test Requirements	Guidance
	B.2.3.b The assessor shall examine all relevant payment terminal documentation—including payment terminal vendor security guidance/policy—necessary to determine which encryption methods are provided by the payment terminal.	
	B.2.3.c The assessor shall examine all relevant software documentation and source code necessary to confirm that the software does not bypass or render ineffective any encryption methods provided by the payment terminal in accordance with the payment terminal vendor's security guidance/policy.	
	B.2.3.d Where the software facilitates encryption of sensitive data, but the payment terminal is not required to provide approved encryption methods (per the *PCI PTS POI Standard*), the assessor shall examine all relevant software documentation and source code necessary to confirm that the encryption methods used or implemented by the software for encrypting sensitive data provide "strong cryptography" and are implemented in accordance with Control Objectives 7.1 and 7.2.	
B.2.4 The software uses only the random number generation function(s) included in the payment terminal's PTS device evaluation for all cryptographic operations involving sensitive data or sensitive functions where random values are required and does not implement its own random number generation function(s).	**B.2.4.a** The assessor shall examine all relevant software documentation and source code necessary to determine whether the software requires random values to be generated for any cryptographic operations involving sensitive data or sensitive functions.	The unpredictability of random numbers is of critical importance to ensure the effectiveness of cryptographic operations. It is not a trivial endeavor to design and implement a secure random number generator. For this reason, the terminal software should only use the random number generation function(s) implemented by the payment terminal for all cryptographic operations involving sensitive data or sensitive functions where random values are required.
	B.2.4.b Where the software requires random values for cryptographic operations involving sensitive data or sensitive functions, the assessor shall examine all relevant payment terminal documentation—including payment terminal vendor security guidance/policy—necessary to determine all of the random number generation functions included in the payment terminal's PTS device evaluation.	

Control Objectives	Test Requirements	Guidance
	B.2.4.c The assessor shall examine all relevant software documentation and source code necessary to confirm that the software uses only the random number generation function(s) included in the payment terminal's PTS device evaluation for all cryptographic functions involving sensitive data or sensitive functions where random values are required and does not implement its own random number generation function(s).	
B.2.5 The software does not facilitate, through its own logical interface(s), the sharing of clear-text account data directly with other software. ***Note:*** *The software is allowed to share clear-text account data directly with the payment terminal's firmware.*	**B.2.5.a** The assessor shall examine all relevant software documentation and source code necessary to determine all logical interfaces of the software, including: • All logical interfaces and the purpose and function of each. • The logical interfaces intended for sharing clear-text account data, such as those used to pass clear-text account data back to the approved firmware of the payment terminal. • The logical interfaces not intended for sharing of clear-text account data, such as those for communication with other software.	Many payment terminals provide mechanisms for the secure reading and exchange of data (SRED). These mechanisms are rigorously tested as part of the payment terminal's PTS device evaluation to confirm that the confidentially and integrity of clear-text account data is maintained during information exchange with the payment terminal's firmware. Software that provides its own mechanisms for sharing clear-text account data directly with other software is more likely to be prone to attacks and the unintended or unauthorized disclosure of clear-text account data than software that uses the payment terminal-provided SRED (or similar) functions.
	B.2.5.b The assessor shall examine all relevant software documentation and source code necessary to confirm that the software does not facilitate sharing of clear-text account data directly with other software through its own logical interfaces.	
	B.2.5.c The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall test the software using all software functions that handle account data to confirm that the software does not facilitate the sharing of clear-text account data directly with other software through its own logical interfaces.	

Control Objectives	Test Requirements	Guidance
B.2.6 The software uses and/or integrates all shared resources securely and in accordance with the payment terminal vendor's security guidance/policy.	**B.2.6.a** The assessor shall examine all relevant software documentation and source code necessary to determine whether and how the software connects to and/or uses any shared resources provided by the payment terminal, and to confirm that: • The software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1 includes detailed instructions for how to configure the software to ensure secure integration with shared resources. • The software vendor's implementation guidance for secure integration with such shared resources is in accordance with the payment terminal vendor's security guidance/policy.	Where the software uses or integrates shared resources provided by the payment terminal, the software must use or integrate resources in accordance with the payment terminal vendor's guidance/policy. Failure to use such shared resources in accordance with payment terminal guidance puts any sensitive data shared with such resources at greater risk of unauthorized disclosure.
	B.2.6.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall test the software using all software functions that use or integrate shared resources to confirm that any connections to or use of shared resources are handled securely.	

Control Objectives	Test Requirements	Guidance
B.2.7 The software does not bypass or render ineffective any application segregation enforced by the payment terminal.	**B.2.7.a** The assessor shall examine all relevant payment terminal documentation—including the payment terminal vendor's security guidance/policy—necessary to determine whether and how application segregation is enforced by the payment terminal.	Many payment terminals enforce logical separation between software applications. In the context of this module, software applications are logical entities that do not meet the PTS definition of "firmware". Logical application segmentation controls are intended to prevent one application on the payment terminal from interfering or tampering with other applications. However, these logical segregation controls are not intended to prevent applications from sharing data. They are mainly intended to prevent applications from modifying the structure or function of other applications or the payment terminal's firmware. To preserve the integrity of payment terminal application-segregation controls, all terminal software should adhere to those segregation controls and not include or introduce any function(s) that would allow the software to be used (intentionally or unintentionally) to bypass or defeat device-level application segregation.
	B.2.7.b The assessor shall examine all relevant software documentation and source code necessary to confirm that the software does not introduce any function(s) that would allow it to bypass or defeat any device-level application segregation controls.	
B.2.8 All software files are cryptographically signed to facilitate cryptographic authentication of the software files by the payment terminal firmware.	**B.2.8.a** The assessor shall examine the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1 to confirm it includes detailed instructions for how to cryptographically sign the software files in a manner that facilitates the cryptographic authentication of all such files by the payment terminal.	To support cryptographic authentication of software files by the payment terminal, software vendors must cryptographically "sign" all of the software files (including all binaries, libraries, and configuration files) using digital certificates where the payment terminal vendor is included in the certificate chain. Additionally, the cryptographic signing process should incorporate the use of a secure cryptographic device (SCD), typically provided by the payment terminal vendor. Cryptographic signing should also be performed under dual control to protect the integrity of all cryptographic keys, software files, and the cryptographic signing process in general.
	B.2.8.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall confirm that all software files are cryptographically signed in a manner that facilitates the cryptographic authentication of all software files.	

Control Objectives	Test Requirements	Guidance
	B.2.8.c Where the software supports the loading of files outside of the base software package(s), the assessor shall determine whether each of those files is cryptographically signed in a manner that facilitates the cryptographic authentication of those files by the payment terminal. For any files that cannot be cryptographically signed, the assessor shall justify why the inability to cryptographically sign each such files does not adversely affect the security of the software or the underlying payment terminal.	
	B.2.8.d The assessor shall examine all relevant software documentation and source code necessary to determine whether and how the software supports EMV® payment transactions. Where EMV payment transactions are supported by the software, the assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall confirm that all EMV Certification Authority Public Keys are cryptographically signed in a manner that facilitates the cryptographic authentication of those files by the payment terminal.	Where terminal software supports EMV payment transactions, the EMV Certificate Authority public keys should also be signed and cryptographically authenticated using the same methods and procedures as the terminal software files.
B.2.9 The integrity of software prompt files is protected in accordance with Control Objective B.2.8.	**B.2.9.a** The assessor shall examine all relevant software documentation and source code necessary to determine whether the software supports the use of data entry prompts and/or prompt files. Where the software supports such features, the assessor shall confirm the software protects the integrity of those prompts as defined in Test Requirements B.2.9.b through B.2.9.c.	How sensitive data (including PIN and other account data) is captured and handled by the software and underlying payment terminal is often controlled using prompt files. Prompt files are configuration files that control software display prompts. To preserve the integrity of the prompts, prompt files should be stored and managed securely. Anywhere clear-text data entry is facilitated by the software, prompt controls should be implemented. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	B.2.9.b The assessor shall examine the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1 to confirm it includes detailed instructions for directing software stakeholders to cryptographically sign all prompt files in a manner that facilitates the cryptographic authentication of all such files in accordance with B.2.8.	Many prompt files are stored within a secure boundary of the device, such as a Secure Chip or Secure Element or within a Trusted Execution Environment. When prompt files are to be maintained in shared storage locations, the files should be cryptographically signed and authenticated by the payment terminal prior to installation or execution.
	B.2.9.c The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall confirm that all prompt files are cryptographically signed in a manner that facilitates the cryptographic authentication of those files by the payment terminal in accordance with B.2.8.	

Control Objectives	Test Requirements	Guidance
Control Objective B.3: Terminal Software Attack Mitigation Software security controls are implemented to mitigate software attacks.		
B.3.1 The software validates all user and other external inputs. ***Note:*** *Control Objectives B.3.1 through* B.3.3 *are extensions of Control Objective 4.2. Validation of these control objectives should be performed at the same time.*	**B.3.1.a** The assessor shall examine all relevant software documentation and source code necessary to identify all external inputs to the software. For each user or other external input, the assessor shall examine all relevant software documentation and source code to confirm that inputs conform to a list of expected characteristics and that all input that does not conform to expected characteristics is rejected by the software or otherwise handled securely.	Any terminal software functions that accept externally supplied data (directly or indirectly) is a potential attack vector, particularly when that data is evaluated by a command interpreter. Injection attacks are common for almost all types of software and are intended to manipulate input data in a way that causes software to behave unexpectedly or unintentionally. For example, software that accepts externally supplied information, such as a file name or file path to construct a search command, can be easily manipulated to disclose information about sensitive files and resources that were never intended to be accessed through the software interface. To protect against this and other types of injection attacks, all input data should be validated (or filtered, sanitized, etc.) before the information is sent to any command interpreter. Inputs for terminal software tend to involve simple commands and data. Therefore, all terminal software input data should be validated against a defined and restricted set of acceptable values before passing the data to any command interpreter. Any data that is not explicitly identified as an acceptable value or an acceptable range of values should be rejected.
	B.3.1.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall test the software by attempting to supply each user or other external input with invalid or unexpected characteristics to confirm that the software validates all inputs and either rejects or securely handles all unexpected characteristics.	
B.3.1.1 All string values are validated by the software.	**B.3.1.1.a** The assessor shall examine all relevant software documentation and source code necessary to identify all terminal software functions where string values are passed as inputs to confirm that all strings are checked for text or data that can be erroneously or maliciously interpreted as a command.	Externally supplied inputs that can be interpreted as commands are particularly susceptible to injection attacks. Even if externally supplied inputs are processed or transformed in some way (e.g., augmented with additional data or sanitized), they may still be susceptible. *(continued on next page)*

Control Objectives	Test Requirements	Guidance
	B.3.1.1.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.), the assessor shall test the software by attempting to supply each of the identified functions with data that includes commands to confirm that the software either rejects such inputs or otherwise handles such inputs securely.	Therefore, all inputs that can be interpreted as commands must be handled securely so that the execution of any constructed commands is controlled, as opposed to blindly executing whatever commands are included in the string.
B.3.1.2 The software checks inputs and rejects or otherwise securely handles any inputs that violate buffer size or other memory allocation thresholds.	**B.3.1.2.a** The assessor shall examine all relevant software documentation and source code necessary to identify all software functions that handle buffers and process data supplied by external inputs. For each of the noted functions, the assessor shall confirm that each of the identified functions: • Uses only unsigned variables to define buffer sizes. • Conducts checks that confirm that buffers are sized appropriately for the data they are intended to handle, including consideration for underflows and overflows. • Rejects or otherwise securely handles any inputs that violate buffer size or other memory allocation thresholds.	Payment terminals and terminal software often leverage low-level programming languages, such as C and C++. These languages allow the software to directly manipulate OS-level or hardware-level features and functions. Using low-level programming languages offers many benefits but also has several drawbacks. Low-level programming languages are susceptible to attacks that use low-level characteristics to manipulate the software or the underlying hardware. Buffer overflows and underflows are examples of these types of attacks. To protect against buffer overflow attacks, all terminal software functions that define or control buffer sizes should compare the amount of data intended for those buffers with the buffer size. Data that violates buffer size thresholds (overflows and underflows) should be rejected or otherwise handled securely.
	B.3.1.2.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.) the assessor shall test the software by attempting to supply each noted function with inputs that violate buffer size thresholds to confirm that the software either rejects or securely handles all such attempts.	

Control Objectives	Test Requirements	Guidance
B.3.2 Return values are checked, and error conditions are handled securely.	**B.3.2.a** The assessor shall examine all relevant software documentation and source code necessary to identify all software functions that handle the sensitive data predefined in Control Objective 1.2. For each of the noted software functions, the assessor shall confirm that each function: • Checks return values for the presence of sensitive data • Processes the return values in a way that does not inadvertently "leak" sensitive data.	Another common technique used by attackers to compromise sensitive data that is stored, processed, or transmitted by software is to manipulate the software in a way that generates unhandled exceptions. Unhandled exceptions are error conditions that the software vendor has not anticipated and, therefore, has not factored into the software design. If an attacker can manipulate a software function that is known to handle sensitive data in a way that generates a condition that the software does not handle properly, it is possible that the software may output an error that includes sensitive data. To protect against attacks involving unhandled exceptions, all terminal software functions handling sensitive data should include processes or routines that instruct the software how to treat unknown exceptions. These processes should determine what information to include in any error codes or values. The disclosure of sensitive data through error conditions or error reporting, whether intentional or accidental, should be avoided.
	B.3.2.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.), the assessor shall test each software function that handles sensitive data by attempting to manipulate the software in a manner that generates an unhandled exception to confirm that error conditions do not expose sensitive data.	
B.3.3 Race conditions are avoided.	**B.3.3.a** The assessor shall examine all relevant software documentation and source code necessary to identify all software functions that rely on synchronous processing. For each of the noted functions, the assessor shall confirm that protection mechanisms have been implemented in the software to mitigate race conditions.	Race conditions can arise when the software requires sequential processing of data to perform some software function. For example, a "time-of-use, time-of-check race condition" exists when a file is checked at one point and used immediately after, with the assumption that the previous check is still valid. This assumption may not be correct if the system allows the file to be modified in between. If an attacker can identify and manipulate the software to take advantage of a race condition, they may be able to execute arbitrary code or generate other conditions that the attacker could exploit further.

Control Objectives	Test Requirements	Guidance
	B.3.3.b The assessor shall install and configure the software in accordance with the software vendor's implementation guidance required in Control Objectives 12.1 and B.5.1. Using an appropriate "test platform" and suitable forensic tools and/or methods (commercial tools, scripts, etc.), the assessor shall test each software function that relies on synchronous processing by attempting to generate a race condition (such as through specially-crafted attacks intended to exploit the timing of synchronous events) to confirm that the software is resistant to such attacks.	To protect against race conditions, protection mechanisms should be implemented by the terminal software to control sequential processing more tightly. Using the example described above, a "locking" mechanism could be used to prevent updates to the file until the file can be processed completely. Regardless of the methods used, any terminal software that requires sequential processing of data for its operation should implement protections to avoid race conditions.

Control Objectives	Test Requirements	Guidance
Control Objective B.4: Terminal Software Security Testing The software is tested rigorously for vulnerabilities prior to each release.		
B.4.1 A documented process is maintained and followed for testing software for vulnerabilities prior to each update or release. ***Note:*** *This control objective is an extension of Control Objective 10.2. Validation of these control objectives should be performed at the same time.*	**B.4.1.a** The assessor shall examine all relevant software documentation and evidence necessary to confirm that the software vendor maintains a documented process in accordance with Control Objective 10.2 for testing the software for vulnerabilities prior to each update or release, and that the documented process includes detailed descriptions of how the vendor tests for the following: • The presence or use of any unnecessary ports and protocols. • The unintended storage, transmission, or output of any clear-text account data. • The presence of any default user accounts with default or static access credentials. • The presence of any hard-coded authentication credentials in code or in configuration files. • The presence of any test data or test accounts. • The presence of any faulty or ineffective software security controls.	Many software vulnerabilities are the result of the software vendor's failure to remove test functions or data. These lingering functions and data can provide an attacker with a path to compromise the software. Before software is released to the public, it must be tested to confirm that test functions and data are not included in the release version. Examples of such functions and data that must be explicitly removed prior to release include: • Any communication ports or protocols that are not absolutely required for software operation. • Any functions that allow the unintended storage, transmission, or output of any clear-text account data. • Any hard-coded authentication credentials in code or configuration files. • Any test data or test user accounts. • Any faulty or ineffective software security controls and protection mechanisms.
	B.4.1.b The assessor shall examine all relevant documentation and evidence necessary (such as software testing artifacts, etc.) to confirm that the software is tested for vulnerabilities prior to each release and that the testing covers the following: • The presence or use of any unnecessary ports and protocols. • The unintended storage, transmission, or output of any clear-text account data. *(continued on next page)*	

Control Objectives	Test Requirements	Guidance
	B.4.1.b • The unintended storage, transmission, or output of any clear-text account data. • The presence of any default user accounts with static access credentials. • The presence of any hard-coded authentication credentials in code or in configuration files. • The presence of any test data or test accounts. • The presence of any faulty or ineffective software security controls.	

Control Objectives	Test Requirements	Guidance
Control Objective B.5: Terminal Software Implementation Guidance The software vendor provides stakeholders with clear and thorough guidance on the secure implementation, configuration, and operation of the software on applicable payment terminals.		
B.5.1 The software vendor provides implementation guidance on how to implement and operate the software securely for the payment terminals on which it is to be deployed. ***Note**: This control objective is an extension of Control Objective 12.1. Validation of these control objectives should be performed at the same time.*	**B.5.1** The assessor shall examine all relevant software documentation and evidence necessary to confirm that the software vendor provides detailed implementation guidance to stakeholders in accordance with Control Objective 12.1 on how to securely implement and operate the software for all applicable payment terminals.	Because many security features used by terminal software are provided by the underlying payment terminal, the terminal software vendor should include instructions in its implementation guidance on how to configure all the available security features of both the terminal software and underlying payment terminal where applicable.
B.5.1.1 Implementation guidance includes detailed instructions for how to configure all available security options and parameters of the software.	**B.5.1.1** The assessor shall examine software vendor implementation guidance to confirm it includes detailed instructions on how to configure all available security options and parameters of the software in accordance with Control Objective B.1.3.	
B.5.1.2 Implementation guidance includes detailed instructions for how to securely configure the software to use the security features and functions of the payment terminal where applicable.	**B.5.1.2** The assessor shall examine the software vendor implementation guidance to confirm it includes detailed instructions on how to securely configure the software to use the security features and functions of the payment terminal where applicable.	
B.5.1.3 Implementation guidance includes detailed instructions for how to configure the software to securely integrate or use any shared resources provided by the payment terminal.	**B.5.1.3** The assessor shall examine the software vendor implementation guidance to confirm it includes detailed instructions on how to configure the software to securely integrate or use any shared resources provided by the payment terminal in accordance with Control Objective B.2.6.	

Control Objectives	Test Requirements	Guidance
B.5.1.4 Implementation guidance includes detailed instructions on how to cryptographically sign the software files in a manner that facilitates the cryptographic authentication of all such files by the payment terminal.	**B.5.1.4** The assessor shall examine the software vendor implementation guidance to confirm it includes detailed instructions on how to cryptographically sign the software files in a manner that facilitates the cryptographic authentication of all such files by the payment terminal in accordance with Control Objective B.2.8.	
B.5.1.5 Implementation guidance includes instructions for stakeholders to cryptographically sign all prompt files.	**B.5.1.5** The assessor shall examine the software vendor implementation guidance to confirm it includes detailed instructions for stakeholders to cryptographically sign all prompt files in accordance with Control Objective B.2.9.	
B.5.2 Implementation guidance adheres to payment terminal vendor guidance on the secure configuration of the payment terminal.	**B.5.2** The assessor shall examine the payment terminal vendor's security guidance/policy and the software implementation guidance required in Control Objective B.5.1 to confirm that the software implementation guidance aligns with the payment terminal vendor's security guidance/policy.	Software implementation guidance must exclude instructions that conflict with the guidance and recommendations of the payment terminal vendor. Software implementation guidance must align with the payment terminal vendor's security guidance/policy. Otherwise, software users who rely on the software vendor for instructions may unknowingly configure the software and/or the underlying payment terminal improperly.