



INFORMATION SUPPLEMENT

Payment Page Security and Preventing E-Skimming - Guidance for PCI DSS Requirements 6.4.3 and 11.6.1

Version: 1.0

Date: March 2025

Author: PCI Security Standards Council

Contents

Introduction	1
Intended Use and Target Audience	1
Terminology	2
Threats and Risks with Modern E-Commerce Websites	4
How Scripts Can Be Compromised	5
<i>Types of Skimming Attacks</i>	5
<i>Silent Skimming</i>	5
<i>Double-Entry Skimming</i>	6
PCI DSS Requirements 6.4.3 and 11.6.1 Explained	7
PCI DSS Requirement 6.4.3	7
<i>Authorization</i>	7
<i>Integrity</i>	7
<i>Inventory and Justification</i>	8
PCI DSS Requirement 11.6.1.....	8
<i>What May Indicate an Unauthorized Modification?</i>	8
<i>Evaluating HTTP Headers and the Script Contents of Payment Pages</i>	8
<i>Frequency of Evaluation</i>	9
Common Payment Page Scenarios	10
Merchant-Hosted Payment Forms	10
Embedded Payment Forms (iframes)	10
Redirection Mechanisms	11
Fully Outsourced Merchant Website	11
Requirements 6.4.3 and 11.6.1 – Applicability and Responsibilities	12
PCI DSS Scoping for E-commerce Environments	14
Multi-page Applications	14
Single-Page Applications.....	14
Approaches for Managing Third-Party Scripts	15
Best Practices for Minimizing Risk When Using Third-Party Scripts	16
Introduction	16
Minimize the Number of Scripts	16
<i>Move Scripts to Isolated Iframes</i>	16
<i>Limit Script Sources</i>	16
<i>Understand and Baseline Script Behavior</i>	16
<i>Technical Security Assessments</i>	16
Controls and Techniques to Help Meet Requirements 6.4.3 and 11.6.1	17
Details: Controls Highlighted in Table 4.....	17
Content Security Policy (CSP)	18
Sub-resource Integrity (SRI).....	18

Webpage Monitoring	19
Proxy-based Solutions	19
Details: Techniques Highlighted in Table 4	20
<i>File Hashing</i>	20
<i>Limiting Sources by URL</i>	20
<i>Nonces</i>	20
<i>Integrating Script Inventory into the Development Lifecycle</i>	20
<i>Manual Processes</i>	20
<i>Behavior Monitoring</i>	20
<i>Static Analysis Script Monitoring</i>	21
<i>Tamper-Resistant Scripts</i>	21
Use of Compensating Controls	21
Use of the Customized Approach	21
How Third-Party Service Providers Can Help	22
Best Practices for All Merchants	22
Best Practices for Small Merchants	22
Best Practices for Medium and Large Merchants	23
Assessment Evidence for Script Management	24
Testing Procedures and Related Assessment Evidence for Requirement 6.4.3	24
More information about evidence for Requirement 6.4.3	26
Testing Procedures and Related Assessment Evidence for Requirement 11.6.1	27
More Information About Evidence for Requirement 11.6.1	29
Further Resources and Guidance	29
About the PCI Security Standards Council	29

Introduction

Data breaches that occur during e-commerce transactions, referred to as e-skimming, have increased significantly in recent years. As e-commerce platforms become more complex, businesses have become more reliant on external scripts, making these types of e-skimming attacks more common. Scripts running in consumers' browsers are now a significant target for attackers trying to steal payment card data.

E-commerce platforms rely on various scripts to deliver functionality, enhance user experience, and support business analytics. These types of script languages are programming languages that run in a web browser (like Chrome, Safari, and Edge), and have changed the web from static pages that display images to the more dynamic computing environment in use today.

To address this risk, *PCI DSS v4.x* includes Requirements 6.4.3 and 11.6.1, which aim to reduce the risk during e-commerce transactions by ensuring payment page scripts are authorized, checked for integrity, and monitored for tampering. These requirements address the need to manage both scripts and security-impacting HTTP headers carefully to avoid unauthorized changes to webpages either in the server environment or as rendered or executed in the consumer's browser.

Intended Use and Target Audience

This guidance is intended for any entity that processes payment card transactions through e-commerce or with a webpage that can impact the security of e-commerce payments by using embedded iframes.

This document provides guidance for merchants and third-party service providers (TPSPs) to take steps towards meeting PCI DSS Requirements 6.4.3 and 11.6.1.

The level of detail provided in this document is based on the assumption that the core target audience has a general knowledge of the payment card industry, as well as a foundational understanding of:

- E-commerce payment transaction flows.
- Basic e-commerce website infrastructure.

As such, the target audience for this document includes, but is not limited to:

- Merchants, service providers, third-party service providers (TPSPs), and others that implement controls to meet PCI DSS requirements for their enterprises or clients.
- Assessors (such as Qualified Security Assessors (QSAs) and Internal Security Assessors (ISAs)) that confirm that PCI DSS requirements are being met.
- PCI Forensic Investigators (PFIs) that determine whether PCI DSS requirements have been implemented as part of an investigation.
- Organizations that manage compliance programs (such as acquirers, payment brands, or other entities) that are responsible for determining and confirming their clients' PCI DSS validation efforts. These organizations are also called compliance-accepting entities.

The PCI Security Standards Council (PCI SSC) is not responsible for enforcing compliance or determining whether a particular implementation is compliant. Entities should work with the organization(s) that manages their compliance program, such as an acquirer (merchant bank), payment brand, or other entity, to understand the entity's compliance validation and reporting responsibilities.

Note: This information supplement provides supplemental guidance that does not add, extend, replace, or supersede requirements in any PCI SSC standard. At the time of publication, the current version of PCI DSS is v4.0.1; however, the general principles and practices offered here may also apply to future versions.

Terminology

The following terms and acronyms are used throughout this document:

- **E-commerce system(s)** – the system(s) within a merchant’s webpage that either redirects consumers from their website to a TPSP/payment processor or that includes a TPSP’s/payment processor/s embedded payment page or form (for example, one or more iframes).
- **Inline Frame (iframe)** – In the context of this document, an iframe embeds an HTML element (the payment page) within a non-payment page (the parent page), inserting the payment page onto the non-payment page. The iframe acts as a “window” that shows the content from the payment page onto the non-payment page.
- **Non-payment page** – In the context of this document, any other page that embeds the payment form and form fields. For example, the page that displays the iframe. Also referred to as a parent page in this document.
- **Payment form** - The specific input fields used for entering payment account data, whereas a payment page encompasses the entire webpage where the form resides
- **Payment page** – According to the PCI DSS Glossary¹, “a web-based user interface containing one or more form elements intended to capture account data from a consumer or submit captured account data for purposes of processing and authorizing payment transactions. The page can be rendered as any one of the following:
 - A single document or instance.
 - A document or component displayed in an inline frame (iframe) within a non-payment page.
 - Multiple documents or components each containing one or more form elements contained in multiple iframes within a non-payment page.

The payment page is also referred to as a child page in this document when it is included in an iframe.

- **Payment page script** – According to the PCI DSS Glossary, “Any programming language commands or instructions on a payment page that are processed and/or interpreted by a consumer’s browser, including commands or instructions that interact with a page’s document object model (DOM). Examples of programming languages are JavaScript and VB Script; neither markup-languages (for example, HTML) or style-rules (for example, CSS) are programming languages.” Another example of a programming language that might be used for payment page scripts is Web Assembly (WASM).

Payment page scripts include scripts related to advertising, product selection, page security or structure, and more. Historically, the emphasis has often been on the web servers themselves, but now it is equally important to review how applications are designed and delivered to the consumer’s browser.

- **Security Impacting HTTP Headers** – Examples of security-impacting headers may include:
 - Content Security Policy (CSP)
 - X-Frame-Options (protection against clickjacking)
 - Strict Transport Security (HSTS)
 - X-XSS-Protection (XSS Filter)
 - X-Content-Type-Options (prevent MIME sniffing)
 - Set-Cookie

¹ PCI DSS Appendix G PCI DSS Glossary of Terms, Abbreviations, and Acronyms.

- Access-Control-Allow-Origin (cross-origin requests)
- Referrer-Policy
- Permissions-Policy
- Cross-Origin-Opener-Policy / Cross-Origin-Embedder-Policy / Cross-Origin-Resource-Policy

Which headers matter depends on a website's security architecture and supported browser versions. For more details, see the OWASP Secure Headers Project and the World Wide Web Consortium which publishes various header standards ².

- **Software Development Kit (SDK)** – a set of tools for third-party developers to use in producing applications using a particular framework or platform.

Definitions for other payment-specific terms noted within this document can be found in the *Payment Card Industry Data Security Standard (PCI DSS) v4.x, Appendix G PCI DSS Glossary of Terms, Abbreviations, and Acronyms*.

² <https://owasp.org/www-project-secure-headers/> and <https://www.w3.org/TR/>

Threats and Risks with Modern E-Commerce Websites

When scripts are not correctly managed—particularly those sourced from third parties—they can lead to data leakage and theft through e-skimming attacks.

Scripts may be loaded from:

- First-party servers controlled by the entity (for example, the merchant).
- Third-party servers, such as those provided by third-party service providers that can, in turn, load scripts from additional parties (fourth-party servers and beyond).

Figure 1. Components of an E-Commerce Webpage

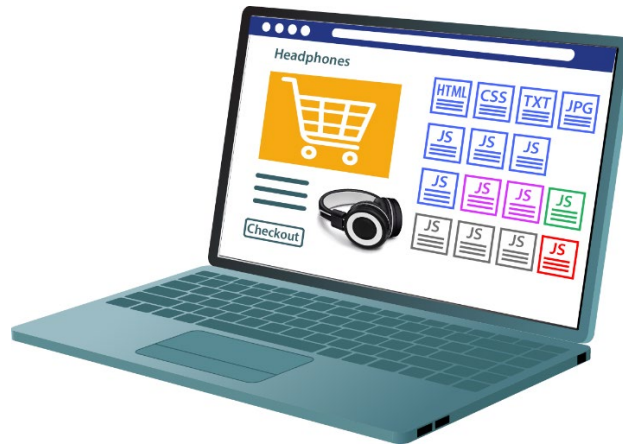
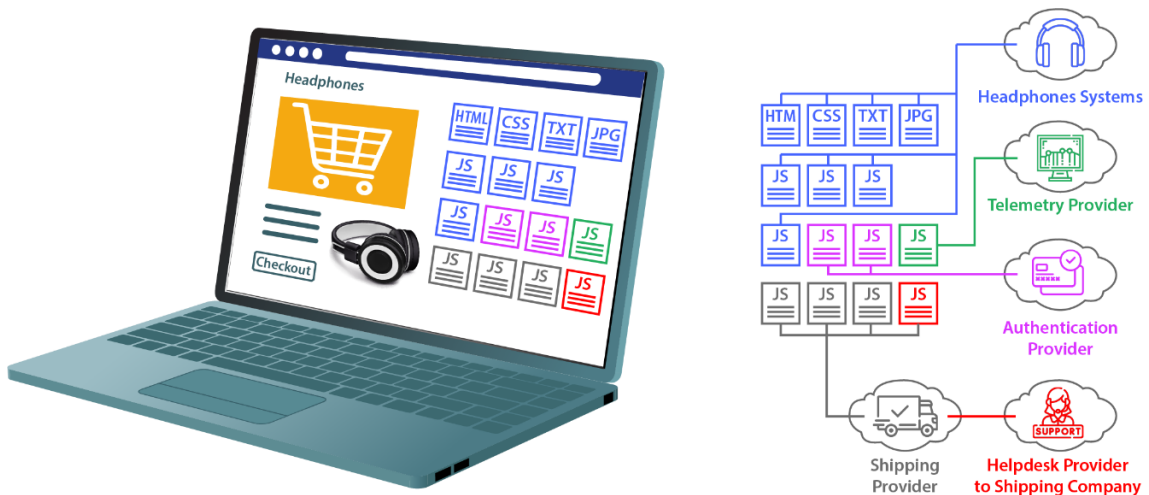


Figure 2. Third-party Components of an E-Commerce Webpage



In some cases, scripts have the ability to interact with sensitive portions of a website and can inadvertently create openings for attackers to exploit if the scripts are not authorized or monitored. For example, third-party scripts used for analytics, marketing, usage tracking, tag management, form validation, or payment processing might be manipulated by malicious actors to capture payment data.

How Scripts Can Be Compromised

Types of attack scenarios include:

- **Supply-chain attacks:** Many e-commerce websites rely on scripts from third-party service providers (TPSP) for functions. If attackers compromise those third-party scripts, they can insert malicious code without the TPSP or merchant immediately noticing.
- **Script injection attacks:** Attackers inject unauthorized scripts into the payment page. The modified scripts capture payment details and transmit them to attacker-controlled domains.

These types of attacks are also called Magecart or Formjacking. Because modern websites rely heavily on scripting for their interactive capabilities, attackers can potentially exploit any script that runs in a consumer's web browser to skim payment data.

By proactively managing and monitoring e-commerce scripts, entities can reduce the likelihood of these attacks. While such threats will continue, they can be effectively addressed through well-defined controls—including those specified in PCI DSS Requirements 6.4.3 and 11.6.1—that implement robust script authorization and integrity checks. Focusing on proactive script management enables businesses to maintain the advantages of using scripts while minimizing their exposure to potential security issues.

When a web browser encounters a script, it interprets and runs the instructions in that script. Those instructions can read, alter, or delete other page contents or trigger actions in response to user inputs. For example, a script might display a menu when the consumer taps an icon or let the consumer drag elements around the page. In the context of payments, scripts can:

- Detect when a consumer enters data into a payment card field.
- Recognize when the consumer finishes typing the data and moves to the following field.
- Validate the entered payment card data and provide feedback to the consumer if it fails. For example, the script could:
 - Change the payment card field to indicate a problem (for example, change the border of the input field to red).
 - Display a warning message that the data was entered incorrectly.
 - Disable the submit button so the incorrect data cannot be submitted.

These capabilities can spot input errors immediately rather than waiting for authorization to fail. However, if the payment page lacks appropriate protections, a malicious script could also capture the payment account data and transmit it to an attacker. As any script running on the page may be able to access and skim payment account data, two new PCI DSS Requirements—6.4.3 and 11.6.1—were introduced in *PCI DSS v4.0* in 2022 to help prevent and detect unauthorized scripts.

Types of Skimming Attacks

There are two main types of e-commerce skimming attacks. While payment account data is skimmed in both types of attacks, the consumer's experience is different in each.

Silent Skimming

In a silent skimming attack, the malicious script operates transparently in the background. Consumers complete their transactions without any indication that their data is being stolen.

Silent skimming attacks can be harder to detect because everything functions as expected from the points of view of both the consumer and the merchant, and, as a result, silent skimming attacks may go undetected for long periods of time.

Double-Entry Skimming

In a double-entry attack, consumers are prompted to enter their card information twice. The attacker inserts or overlays a false payment form before the legitimate form appears to capture the data prior to the intended payment flow. After entering their payment card account data into the false form and encountering an error, consumers are prompted to re-enter the data a second time into the merchant's form. Such attacks rely upon the consumer to go through the payment entry process twice, and because of this are often discovered more quickly—either by consumers reporting the strange behavior or because the merchant notices it while testing their website.

PCI DSS Requirements 6.4.3 and 11.6.1 Explained

PCI DSS Requirement 6.4.3

All payment page scripts that are loaded and executed in the consumer browser must be managed as follows:

- A method is implemented to confirm that each script is authorized.
- A method is implemented to assure the integrity of each script.
- An inventory of all scripts is maintained with written business or technical justification for why each is necessary.

Requirement 6.4.3 has three elements—authorization, integrity, and inventory that includes a technical or business justification—that are designed to prevent skimming attacks before they happen.

According to PCI DSS, Requirement 6.1.1 an entity is expected to document and manage security policies and operational procedures identified in Requirement 6, including those needed to accomplish Requirement 6.4.3. Equally, for PCI DSS Requirement 6.1.2, the entity is expected to document roles and responsibilities for performing all activities in Requirement 6, including those needed to accomplish Requirement 6.4.3.

For merchants using a 3DS solution, validation to PCI DSS Requirement 6.4.3 for 3DS scripts is not required due to the inherent trust relationship between the 3DS service provider and the merchant, as established in the merchant's due diligence and onboarding processes, and the business agreement between the entities.

Any script run outside of the purpose of performing a 3DS functionality is subject to PCI DSS requirement 6.4.3.

Authorization

The first element of Requirement 6.4.3 requires that each script is authorized. The standard does not specify how the method is expected to function, how or who within the entity provides authorization, or whether the process is manual or automated.

Since third-party scripts can be modified without the entity's knowledge, the guidance column for Requirement 6.4.3 clarifies that authorization can occur before a script is added or changed or as soon as possible after a change is made.

Integrity

The second element of Requirement 6.4.3 addresses integrity - ensuring that scripts do not contain unauthorized or malicious content. This implies two checks:

- When introducing a new script
- When updating an existing script

PCI DSS does not mandate a specific mechanism for assuring script integrity; examples of approaches in the guidance column for Requirements 6.4.3 and 11.6.1 include Content Security Policy (CSP) and Sub-Resource Integrity (SRI). However, entities can select alternative solutions that also meet this objective.

See [Content Security Policy \(CSP\)](#) and [Sub-resource Integrity \(SRI\)](#).

An entity can also consider specialized script management tools.

Inventory and Justification

The final element of Requirement 6.4.3 for securing the use of scripts for e-commerce payments is maintaining an inventory of all scripts on the payment page and a business or technical justification for each script's presence. Only including the known and necessary scripts reduces the attack surface available to malicious actors. PCI DSS does not specify how this inventory must be maintained. In simple environments, a spreadsheet might suffice; in more complex ones, entities might rely on automated tools or workflow systems.

PCI DSS Requirement 11.6.1

A change- and tamper-detection mechanism is deployed as follows:

- To alert personnel to unauthorized modification (including indicators of compromise, changes, additions, and deletions) to the security-impacting HTTP headers and the script contents of payment pages as received by the consumer browser.
- The mechanism is configured to evaluate the received HTTP headers and payment pages.
- The mechanism functions are performed:
 - At least weekly.OR
 - Periodically (at a frequency defined in the entity's targeted risk analysis, according to all elements specified in Requirement 12.3.1).

Requirement 11.6.1 recognizes that unauthorized changes might occur. Rather than preventing every unauthorized change outright, the control ensures that if such changes happen, they are recognized and generate alerts so corrective actions can be taken promptly. An unauthorized change often triggers a review of the controls specified in Requirement 6.4.3 (authorization, integrity, inventory, and justification).

The entity is expected to define roles and responsibilities for performing all activities in Requirement 11 (Requirement 11.1.2), including those needed to accomplish Requirement 11.6.1.

Note that there are no policies and procedures identified in Requirement 11.6.1; therefore, this section does include a reference to Requirement 11.1.1 for documentation of policies and procedures related to Requirement 11.6.1.

What May Indicate an Unauthorized Modification?

The change- and tamper-detection mechanism should detect and alert personnel if a security-impacting HTTP header or script:

- Newly appears — A new header or script is introduced (for example, a second CSP directive or a new JavaScript (.js) file).
- Is changed — An existing header or script expected to load on the payment page has been altered or changes behavior from the previously authorized state.
- Is deleted — A header like a CSP directive or a defensive script is removed.

These requirements aim to prevent or minimize theft of payment data. The entity should plan to respond promptly to any alerts received if an entity's mechanism only provides alerts (rather than automatically blocking changes). Rapid response ensures optimal protection for both the entity and its customers.

Evaluating HTTP Headers and the Script Contents of Payment Pages

The tamper-detection mechanism must evaluate both security-impacting HTTP headers and the script content of payment pages. It can be a single or multiple mechanisms that collectively meet the requirement.

Frequency of Evaluation

Requirement 11.6.1 specifies that the mechanism's functions occur at least weekly. Alternatively, an entity may choose to define a targeted risk analysis (TRA) according to PCI DSS Requirement 12.3.1 to determine an alternative frequency that is acceptable to their organization. Whether the organization chooses a weekly schedule or implements an alternative schedule documented in a TRA, it is important to note that malicious modifications could persist without detection between those times, so frequent or even real-time monitoring is recommended where feasible.

Common Payment Page Scenarios

There are different ways that a payment page can be implemented, including:

- Merchant-Hosted Payment Forms
- Embedded Payment Forms (iframes)
- Redirection Mechanisms
- Fully Outsourced Merchant Website

Merchant-Hosted Payment Forms

When the merchant hosts the payment form, the payment data entry fields reside on the merchant's web server, not a payment processor/TPSP website. Payment data may be transmitted to a payment processor/TPSP in different ways, but the payment form itself is always loaded and controlled by the merchant's environment.

Table 1 describes the common architectures for merchant-hosted payments.

Table 1. Common Architectures for Merchant-Hosted Payments

Merchant-Hosted Payment Architecture	Characteristics
Merchant-Posted Payment	<ul style="list-style-type: none"> ▪ Consumer's browser returns payment data to the merchant's website. ▪ Merchant's website submits payment data to the payment processor/TPSP. ▪ All payment data flows through the merchant environment.
Direct-Post Payment	<ul style="list-style-type: none"> ▪ Consumer's browser submits payment data directly to the payment processor/TPSP. ▪ Payment form fields are hosted on merchant website. ▪ Payment data from the consumer does not flow through the merchant environment.


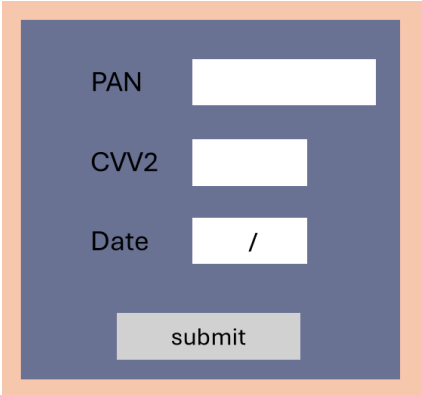
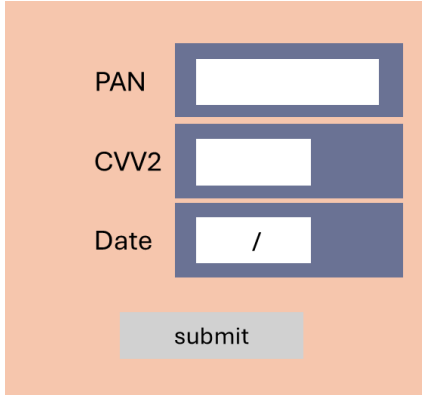
Embedded Payment Forms (iframes)

Iframes are used to embed a payment form within a merchant's website (on the parent page). The payment form may be provided by a TPSP/payment processor or managed directly by the merchant and can be loaded using iframe HTML tags or using JavaScript. Where scripts are used to create an iframe, PCI DSS Requirements 6.4.3 and 11.6.1 will apply to those scripts.

This section uses terms previously defined in the Terminology section. Table 2 provides illustrations and details about the following implementations:

- A single document or instance (a payment page with no iframes)
- A document within a cross-origin iframe
- Multiple documents in separate iframes

Table 2. Iframes Illustrated

Single Document or Instance	Document Within a Cross Origin Iframe	Document Within a Cross Origin Iframe
		
<p>A single webpage containing all input fields for payment card data. Data may be submitted to the entity's server or to a third-party service provider. There are no iframes in this scenario.</p>	<p>An HTML page that contains all payment fields and is rendered in an iframe within a webpage (parent page).</p>	<p>Each field is included in a separate document, and each document is rendered in its iframe within a webpage (parent page).</p>

Redirection Mechanisms

Redirection involves transferring a consumer from the merchant's website to a separate website or a hosted payment page owned by a payment processor/TPSP. The consumer navigates to a new URL, browser window, or popup, as provided by the merchant website.

Redirection mechanisms can be implemented via a server-side configuration setting (for example, HTTP 30x), using HTML tags, or using JavaScript. Where scripts are used as part of a redirection mechanism, PCI DSS Requirements 6.4.3 and 11.6.1 will apply to those scripts.

Fully Outsourced Merchant Website

With a fully outsourced merchant website, the payment processor/TPSP handles all aspects of payment data collection and processing. The merchant's environment does not handle or host any payment forms or scripts that process payment data. Common examples include merchants providing consumers with a payment processor's/TPSP's "link to pay", which consumers use to go directly to the payment processor's/TPSP's website to make the payment.

Requirements 6.4.3 and 11.6.1 – Applicability and Responsibilities

Requirements 6.4.3 and 11.6.1 apply to entities differently depending on how payment solutions are implemented. Using the common payment page scenarios described in *Common Payment Page Scenarios*, this section highlights the applicability of Requirements 6.4.3 and 11.6.1 for the various e-commerce scenarios and clarifies the responsibilities of merchants and TPSPs for payment script security.

Table 3. PCI DSS Requirements 6.4.3 and 11.6.1 – Applicability and Responsibility

Payment Page Scenario	Merchant Responsibility	TPSP Responsibility
Merchant posted payment	Any scripts on the merchant’s webpage(s).	Any scripts the TPSP includes for services they provide.
Direct post payment	Any scripts on the merchant’s webpage(s).	Any scripts the TPSP includes for services they provide.
Embedded payment forms (iframes)	Any scripts on the merchant’s webpage(s) that includes the iframe (i.e., the non-payment page).	Scripts within the iframe(s) provided by the TPSP (i.e., on the payment page).
Redirection mechanisms	Any scripts on the merchant’s webpage(s) that includes the redirection mechanism (i.e., the non-payment page).	Any scripts the TPSP includes for services they provide.
Fully outsourced merchant website	Nothing related to Requirements 6.4.3 and 11.6.1.	Any scripts the TPSP includes for services they provide.

PCI DSS Requirements 6.4.3 and 11.6.1 are included in Self-Assessment Questionnaire (SAQ) A-EP, SAQ D for Merchants, and SAQ D for Service Providers. These requirements are also included in the PCI DSS Report on Compliance (ROC) Template.

SAQ A does not include PCI DSS requirements 6.4.3 or 11.6.1. However, SAQ A does include the following Eligibility Criteria for e-commerce merchants “*The merchant has confirmed that their site is not susceptible to attacks from scripts that could affect the merchant’s e-commerce system(s).*” For information about how an SAQ A merchant can confirm the merchant meets this SAQ A Eligibility Criteria³, refer to *Best Practices for Small Merchants*.

Note that even if a merchant is eligible to complete an SAQ, which SAQ is applicable for one of the above payment page scenarios will vary depending on how the e-commerce environment is implemented, how much involvement the merchant has with implementing and managing the e-commerce environment, whether and for which services a TPSP is used, etc.

Whether an entity is eligible to report the results of their PCI DSS assessment in an SAQ (and which SAQ is appropriate) or must use a ROC to report assessment results is determined by the organizations that manage

³ Refer to the latest version of PCI DSS SAQ A for the complete list of eligibility criteria. Also refer to the FAQ section of the PCI SSC web site at <https://www.pcisecuritystandards.org/> for related FAQs.

compliance programs, such as an acquirer (merchant bank), payment brand, or other entity. Entities should work with those organizations to understand the entity's compliance validation and reporting responsibilities.

PCI DSS Scoping for E-commerce Environments

The scope of PCI DSS requirements, as defined in *PCI DSS v4.x*, section 4 Scope of PCI DSS Requirements, is as follows:

- The cardholder data environment (CDE), which is comprised of:
 - System components, people, and processes that store, process, or transmit cardholder data (CHD) and/or sensitive authentication data (SAD), and
 - System components that may not store, process, or transmit CHD/SAD but have unrestricted connectivity to system components that store, process, or transmit CHD/SAD.

AND

- System components, people, and processes that could impact the security of cardholder data and/or sensitive authentication data.

This includes e-commerce payment pages, and parent pages that are embedding an iframe (whether it is a third-party iframe or an iframe managed by the merchant). Many e-commerce merchants try to reduce the scope of PCI DSS by not storing, processing, and/or transmitting any payment account data on the merchant's webpage. Instead, they rely on third-party service providers (TPSPs) by embedding the TPSP's payment page or by redirecting to a TPSP's page. Note that PCI DSS Requirements 6.4.3 and 11.6.1 do not apply to merchants with webpages that redirect to a TPSP's page (for example, via an HTTP 30x redirect, a meta redirect tag, or a JavaScript redirect).

While embedding a TPSP's payment page in a merchant webpage may reduce the number of applicable PCI DSS requirements for these webpages, embedding a third-party payment page/form does not completely remove the merchant's embedding page (the parent page) from scope—or make Requirements 6.4.3 and 11.6.1 inapplicable. For information about how PCI DSS Requirements 6.4.3 and 11.6.1 apply to different Self-Assessment Questions and the Report on Compliance Template, see [Requirements 6.4.3 and 11.6.1 – Applicability and Responsibilities](#).

Multi-page Applications

Traditionally, e-commerce websites have been multi-page applications: a user navigates to a new URL for each step in the process, and each new page is loaded “fresh.” This approach rebuilds the browser's environment for each page, effectively purging previous scripts from memory. In a multi-page setup, only the parent page where the TPSP payment page is embedded is typically in scope for the merchant's assessment of Requirements 6.4.3 and 11.6.1.

Single-Page Applications

Single-page applications (SPAs) have grown increasingly popular and represent a significant portion of modern e-commerce websites. In an SPA, the browser does not fully reload when the user navigates; instead, JavaScript manipulates the existing page, dynamically adding or removing content. As a result, any scripts loaded during the user's session remain in memory and can continue to operate.

In this scenario, the entire SPA effectively behaves as one continuous “page” from the browser's perspective, including any embedded payment form. Since all scripts are part of the same environment, 6.4.3 and 11.6.1 apply to all pages (or “views”) within that single-page application, which could affect the embedded payment form.

Approaches for Managing Third-Party Scripts

Managing third-party scripts pose a unique challenge because such scripts are often outside the direct control of the entity (for example, a merchant or service provider) that is using third-party scripts. However, these scripts have historically been linked to skimming attacks, so it is important for an entity to be aware of and manage these scripts effectively.

Where a script vendor is not considered a TPSP, it is important that the entity has processes for managing these third-party scripts, including understanding where these scripts are, covering such scripts in the entity's scope, and ensuring that those scripts are managed according to the entity's defined processes for meeting Requirements 6.4.3 and 11.6.1. This section includes tools and solutions that may help manage third-party scripts, as well as best practices that can help meet PCI DSS Requirements 6.4.3 and 11.6.1.

Entities need a mechanism that evaluates both headers and scripts, comparing the current version of the headers and script content with prior known versions to detect deviations. The mechanism can alert without automatically blocking changes, providing flexibility for immediate human intervention. Alternatively, if the mechanism does not block changes in real time, entities should plan for a fast response to any alerts.

Entities can manage third-party scripts in different ways, depending on the entity's technical capabilities and available resources. Common approaches include:

- **Internally Developed Tools:** Custom-built solutions can be used to manage third-party scripts, incorporating script allow listing, integrity verification (for example, with hashing), and other security controls.
- **Commercial and Open-Source Solutions:** Technologies offer monitoring, control, and enforcement of security controls for third-party scripts.
- **Hybrid Approaches:** A combination of internal tools and external scanning tools or services can help detect and respond to anomalies.

These options may be used separately, or as a combined approach to meet PCI DSS requirements 6.4.3 and 11.6.1.

A provider of third-party scripts is not considered a third-party service provider (TPSP) for purposes of PCI DSS Requirements 12.8 and 12.9, if the provider's only service is providing scripts not related to payment processing and where those scripts cannot impact the security of cardholder data and/or sensitive authentication data.

Best Practices for Minimizing Risk When Using Third-Party Scripts

Introduction

Security controls and techniques can help create a comprehensive mechanism for meeting PCI DSS Requirements 6.4.3 and 11.6.1. While the PCI DSS requirements focus on managing and monitoring scripts for unauthorized changes, the controls described in this section can be deployed in various ways to address e-skimming and other threats facing e-commerce websites.

Minimize the Number of Scripts

As noted previously, Requirements 6.4.3 and 11.6.1 focus on payment pages and non-payment pages (or parent pages) that embed or otherwise affect payment pages. Thus, one of the simplest ways to reduce scope is to limit the number of scripts to those strictly necessary. Additional measures include the following:

- *Move Scripts to Isolated Iframes*
- *Limit Script Sources*
- *Understand and Baseline Script Behavior*
- *Technical Security Assessments*

Move Scripts to Isolated Iframes

Using cross-origin or sandboxed iframes, developers can confine scripts to a controlled environment separate from the e-commerce payment and/or parent pages. Properly sandboxed iframes can limit script capabilities, preventing interaction with sensitive payment elements on the main page.

Limit Script Sources

Restricting the URLs from which scripts can be loaded is another effective way to reduce risk. Content Security Policy (CSP) is commonly used to enforce this control through directives like `script-src`, which defines approved sources for JavaScript execution. Alternatively, agent-based webpage monitoring or proxy-based solutions can detect, block, or manage scripts from unapproved sources.

Understand and Baseline Script Behavior

Test scripts for standard behavior in isolation and in their final integrated environment, identifying outputs from scripts and restrict or closely monitor any functionality that is not essential. Consider using application modifications or modern script-control solutions to limit scripts' access to sensitive data.

Technical Security Assessments

Conduct periodic or continuous assessments of third-party scripts to identify vulnerabilities and detect anomalies. These assessments can include static code reviews, behavioral analysis, and/or runtime monitoring.

Controls and Techniques to Help Meet Requirements 6.4.3 and 11.6.1

This section highlights controls and techniques that can be combined to satisfy the elements of Requirements 6.4.3 and 11.6.1 (authorization, inventory, integrity, and alerting). Entities may also leverage these controls and techniques for detecting changes, additions, deletions, and suspicious behaviors (indicators of compromise) in both security-impacting HTTP headers and payment page and non-payment page scripts.

Note that the security controls shown in Table 4 are based on features and security controls included in browsers at the time of publication.

Table 4. Summary of Controls and Techniques

Requirements and Elements	Controls to Help Meet the Elements	Techniques Used with the Control
Requirement 6.4.3		
Authorization	Webpage monitoring, proxy-based, or other authorization methods	-
Integrity	CSP, SRI, webpage monitoring, proxy-based, or other integrity methods	File hashing, limiting scripts by URL, behavior monitoring, static analysis, tamper resistance, or other techniques
Inventory	CSP, webpage monitoring, proxy-based, or other inventory methods (for example, lists)	-
Requirement 11.6.1		
Alerting	CSP, webpage monitoring, proxy-based, or other alerting methods	-
Security-impacting headers	Webpage monitoring, proxy-based, or other methods that alert on changes	-
Script content changes	CSP, webpage monitoring, proxy-based, or other methods that alert on changes	File hashing, behavior monitoring, static analysis, tamper resistance, or other techniques
Script content indicators of compromise	CSP, webpage monitoring, proxy-based, or other methods that alert on indicators of compromise	Limiting scripts by URL, nonces, behavior monitoring, static analysis, tamper resistance, or other techniques
Frequency	CSP, SRI, webpage monitoring, proxy-based, or other scheduling methods	-

Details: Controls Highlighted in Table 4

This section covers several security controls that can detect and, in some cases, prevent unauthorized script activities and generate alerts for relevant stakeholders. These controls support PCI DSS 6.4.3 and 11.6.1 by safeguarding the payment data environment.

Content Security Policy (CSP)

CSP is a native browser feature that allows a web application to set policies dictating how content is loaded and executed. Although CSP can be quite effective for script authorization and, to some extent, script integrity, entities will need to incorporate complementary processes to meet all aspects of Requirements 6.4.3 and 11.6.1.

- Implementation: CSP directives can be delivered via an HTTP response header (for example, Content-Security-Policy) or a meta tag. A CSP can limit script sources (script-src) and define acceptable iframe origins (frame-src). Hash- or nonce-based methods can further enforce integrity by specifically defining the scripts that are permitted to execute on the page.
- Reporting: CSP can be paired with the HTTP [Reporting API](#)⁴ (for example, report-to, report-uri) to capture events where unauthorized scripts are blocked or a policy violation occurs.

Advantages:

- Browser-native and widely supported.
- Can help with script inventory, integrity checks, and partial detection of unauthorized additions or modifications.
- Provides a framework for reporting violations.

Disadvantages:

- The entity will need to implement additional processes for authorization, alerting, tracking header changes, etc. For example, CSP by itself cannot create a list of unauthorized scripts or alert on changes to security -impacting HTTP headers.
- CSP does not maintain a baseline of normal activity. It is static and cannot track historical or expected states across sessions.
- Maintaining a robust CSP (especially with hashes) can be challenging in dynamic environments.
- CSP is not inherently able to detect deletions of security-impacting headers or confirm whether a malicious script changed its internal functionality unless it also contacts disallowed domains.

Sub-resource Integrity (SRI)

By comparing a hash in the HTML tag to the resource the browser loads, SRI helps confirm that certain retrieved resources (like scripts) have not been tampered with. If there is a mismatch, the resource is blocked from execution.

Advantages:

- Straightforward configuration for static scripts.
- Helps ensure script integrity for first-party and some third-party scripts (if updates are predictable and infrequent).

CSP, SRI, and the other security controls summarized in this section can help entities to meet PCI DSS Requirements 6.4.3 and 11.6.1. Each has advantages and disadvantages, and it is up to each entity to decide which controls to implement, and in which combinations. See Table 4 for information about which security controls can help meet the various elements of Requirements 6.4.3 and 11.6.1.

⁴ Refer to the Reporting API, W3C Working Draft, 13 August 2024 at <https://www.w3.org/TR/reporting/>.

Disadvantages:

- SRI is not practical for rapidly changing scripts or content that changes unpredictably (for example, dynamic third-party scripts).
- SRI fails silently—there is no native alert mechanism to inform website owners if a script was blocked.
- SRI does not support reporting.

Webpage Monitoring

Webpage monitoring encompasses any approach that examines a payment page as it is to be rendered in the consumer's browser (or a synthetic environment) to detect malicious or unexpected script activities. These approaches monitor the client side of a running web application, observing how the different scripts interact with the different webpage components, as well as with other browser-based assets (cookies, local storage, etc.). Two models commonly exist:

- **Agent-based:** A monitoring script is injected into the page, tracking document object module (DOM) changes, resource requests, and behavioral patterns.
- **Agentless:** A process or service (for example a headless browser) that regularly navigates through checkout flows, observing loaded scripts, headers, and behaviors without introducing additional scripts in real users' sessions.

Advantages:

- Provides real-time or near-real-time insight into actual script behaviors.
- Can detect unexpected form-overlay attacks, suspicious data exfiltration, etc.

Disadvantages:

- Agent-based solutions require integration into each page; incorrectly implemented solutions could affect performance or conflict with other scripts.
- Agentless solutions may struggle with CAPTCHAs, logins, or state-based flows. They also rely on scheduled checks rather than constant end-user monitoring.

Proxy-based Solutions

Proxy-based solutions intercept traffic at a reverse proxy or content delivery network (CDN) edge, allowing for the modification or analysis of HTML and scripts before they reach the consumer's browser. These solutions can monitor both static and dynamic script references, enforce policies such as CSP, and block unauthorized content in real time.

Advantages:

- Can be deployed with minimal application changes, potentially serving as an agentless solution.
- Provides comprehensive script inventories by monitoring script loads.

Disadvantages:

- May be complex to configure and can introduce latency if not optimized properly.
- Effectiveness depends on detection algorithms, which may generate false positives.

Details: Techniques Highlighted in Table 4

This section covers several techniques included in Table 4 that can be used with the security controls, which are also highlighted in Table 4. For example, limiting sources by URL is a common technique used by CSP (the security control). These controls and techniques can be used to help entities to meet PCI DSS Requirements 6.4.3 and 11.6.1.

- *File Hashing*
- *Limiting Sources by URL*
- *Nonces*
- *Integrating Script Inventory into the Development Lifecycle*
- *Manual Processes*
- *Behavior Monitoring*
- *Static Analysis Script Monitoring*
- *Tamper-Resistant Scripts*

File Hashing

File hashing involves computing a cryptographic hash for a script and verifying that it remains unchanged. This is feasible for static scripts but is impractical if scripts change frequently or are customized per user session.

Limiting Sources by URL

Used in CSP and other controls to implement allow-listing for specific domains. Suspicious domain usage can raise an alert for potential compromise.

Nonces

CSP can authorize scripts via nonces—unique tokens inserted into the script tag and CSP header. If the token does not match, the script is blocked. Nonces can detect unauthorized scripts but do not monitor internal changes within an authorized script. Using nonces with third-party hosted scripts is impractical.

Integrating Script Inventory into the Development Lifecycle

Organizations that incorporate automation into their continuous integration/continuous delivery (CI/CD) pipelines can maintain script inventories by automatically tracking and updating authorized script lists. These processes can include validation checks for both first-party and third-party scripts, ensuring that only approved scripts are deployed. By integrating security assessments into the pipeline, organizations can detect unauthorized changes and ensure the requirements are met before scripts reach production.

Manual Processes

Some requirements (for example, maintaining a script inventory and verifying script authorization) may be handled by manual reviews, sign-offs, or regular website content analysis. Manual steps often complement automated controls.

Behavior Monitoring

Instead of focusing on static script signatures, behavior monitoring checks real-world actions of scripts—for example, if they capture keystrokes, modify or access payment fields, or send data to unknown URLs. An alert or block can be triggered when a script's behavior deviates from its authorized profile.

Static Analysis Script Monitoring

Scripts collected from the website (or from the proxy) are periodically scanned for known indicators of skimming or malicious code patterns. While applicable, static analysis can generate false positives or miss sophisticated obfuscated code.

Tamper-Resistant Scripts

Using compiler tools, scripts can be transformed or instrumented to detect or prevent malicious modifications—whether they occur before runtime or during execution. If tampering is detected, the script can refuse to run or it can raise an alert.

Use of Compensating Controls

Compensating controls are an option if an entity cannot meet a PCI DSS requirement explicitly as stated, due to a legitimate and documented technical or business constraint but has sufficiently mitigated the risk associated with the requirement through the implementation of other, or compensating, controls. On an annual basis, any compensating controls must be documented by the entity, reviewed and validated by the assessor, and included with applicable PCI DSS reporting.

Compensating controls may be an option for an entity to implement to meet either or both Requirements 6.4.3 and 11.6.1.

For information about using, documenting, and assessing compensating controls, refer to *PCI DSS v4.x* Appendices B and C.

Use of the Customized Approach

This approach focuses on a PCI DSS requirement's stated Customized Approach Objective (CAO), allowing entities that decide to implement controls to meet a requirement in a way that does not strictly follow the PCI DSS requirement. The customized approach provides the option for an entity to determine and design the security controls needed to meet the requirement's customized approach objective in a manner unique for that organization, thus providing more flexibility to use alternative security controls (often meaning newer technologies) to meet the CAO.

The Customized Approach Objectives for Requirements 6.4.3 and 11.6.1 are as follows:

- Requirement 6.4.3: Unauthorized code cannot be executed on the payment page as it is rendered in the consumer's browser.
- Requirement 11.6.1: E-commerce skimming code or techniques cannot be added to payment pages as received by the consumer browser without a timely alert being generated. Anti-skimming measures cannot be removed from payment pages without a prompt alert being generated.

Qualified Security Assessors (QSAs) or Internal Security Assessors (ISAs) evaluate the entity's approach and determine whether the entity meets all Customized Approach documentation and evidence requirements.

Note that entities completing a Self-Assessment Questionnaire (SAQ) are not eligible to use a customized approach; however, these entities may elect to have a QSA or ISA perform their assessment and document it in a Report on Compliance (ROC) Template. Questions about using a customized approach should be referred to the organization (typically, payment brands or acquirers) that accepts the entity's compliance documents. Compliance documents are SAQs, Attestations of Compliance (AOCs), and ROCs.

For information about using, documenting, and assessing a customized approach, refer to *PCI DSS v4.x* introductory section 8 and Appendix D.

How Third-Party Service Providers Can Help

TPSPs offering e-commerce or payment services might assist merchants in meeting Requirement 6.4.3 and 11.6.1 by:

- **Hosting the Payment Page:** Shifting responsibility for these controls to the TPSP for all code within the payment page.
- **Supplying Secure SDKs:** Embedding protective measures directly into merchant implementations and providing monitoring or blocking features using a tamper-resistant script(s).
- **Providing Monitoring Services:** Tracking e-skimming indicators on behalf of the merchant's integrated payment flows.
- **Recommending Best Practices:** Leveraging their technical knowledge to guide merchants in securely embedding payment forms.

Merchants should confirm which of the TPSP's services were covered by the TPSP's PCI DSS assessment—this should be evidenced in the TPSP's PCI Attestation of Compliance (AOC).

TPSPs are expected to meet PCI DSS Requirement 12.9.1—to provide customers with written agreements and an acknowledgement that the TPSPs is responsible for the security of account data. TPSPs are also expected to meet PCI DSS Requirement 12.9.2—to provide customers with information about the TPSP's PCI DSS compliance status and about which PCI DSS requirements are the responsibility of the TPSP, which are the responsibility of the customer, and any requirements where responsibilities are shared between the customer and the TPSP. These responsibilities should be provided by the TPSP in a clear format that includes how, or if, each party is expected to address Requirements 6.4.3 and 11.6.1.

Note that TPSPs offering e-commerce or payment services cannot fully address skimming risks on the merchant's website, given that they do not fully control the merchant's website. Merchants are encouraged to implement robust mechanisms to evaluate and control the scripts that merchants allow to be executed from merchant webpages.

The following sections describe best practices for various sized merchants working with third-party service providers:

Best Practices for All Merchants

Evaluate third-party service providers (TPSPs) to ensure that the TPSPs are providing sufficient technology and controls to secure the payment processing the TPSP(s) provides. Merchants should also work with their TPSP(s) to determine the best ways in which to manage and launch the TPSP payment pages to minimize the scope of the PCI DSS e-commerce script requirements. Additionally, merchants should work with their TPSP to obtain guidance about how to implement the TPSP's solution securely.

Best Practices for Small Merchants

Work with TPSPs/payment processor(s) that provide an embedded payment page/form (for example, one or more iframes) to get confirmation that, when implemented according to instructions from the TPSP/payment processor, the TPSP's/payment processor's solution includes techniques that protect the merchant's payment page from script attacks. Alternatively, small merchants can elect to implement techniques such as those detailed in PCI DSS Requirements 6.4.3 and 11.6.1 to protect the merchant's webpage from scripts targeting account data; these techniques may be implemented by the merchant or by a third party.

Best Practices for Medium and Large Merchants

Consider the technologies being used for managing scripts and determine how best to develop webpages to limit the potential for script-based payment attacks. Implement mechanisms to evaluate and control the scripts executed from webpages.

Assessment Evidence for Script Management

PCI DSS includes testing procedures for each requirement and in the Report on Compliance (ROC) Template. In addition, each Self-Assessment Questionnaire (SAQ) includes an Expected Testing column for entities to document which evidence was reviewed and the results of that review from testing for each requirement.

This section addresses assessment evidence that:

- An entity should have available for the entity's PCI DSS assessment to show that Requirements 6.4.3 and 11.6.1 are being met.
- An assessor should expect to receive as part of a PCI DSS assessment.

The assessed entity should be prepared to either:

- Provide the evidence to their assessor or,
- Review the evidence themselves if they are conducting a self-assessment (for example, documented in an SAQ).

Testing Procedures and Related Assessment Evidence for Requirement 6.4.3

The tables in this section address the intended purpose of each testing procedure and the evidence that an assessed entity is expected to have available and provide during a PCI DSS assessment of Requirement 6.4.3.

The following tables are included:

- Table 5 summarizes the purpose of the testing procedures and the expected evidence for Requirement 6.1.1, for security policies and operational procedures identified in Requirement 6. This table only includes information related to security policies and operational procedures for Requirement 6.4.3
- Table 6 summarizes the purpose of the testing procedures and the expected evidence for Requirement 6.1.2, which governs the roles and responsibilities that need to be defined to perform activities in Requirement 6. This table only includes information related to roles and responsibilities for Requirement 6.4.3.
- Table 7 summarizes the purpose of the testing procedures and the expected evidence for Requirement 6.4.3.

Table 5. Testing Procedures and Expected Evidence for Requirement 6.1.1 for Security Policies and Operational Procedures to Support Requirement 6.4.3

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 6.1.1	
All security policies and operational procedures that are identified in Requirement 6 are:	
<ul style="list-style-type: none"> ▪ Documented ▪ Kept up to date ▪ In use ▪ Known to all affected parties 	
Testing Procedure Purpose	Expected Evidence from Entity
6.1.1 Security policies and operational procedures are documented, updated as needed, and in use.	<p>Documented policies and procedures with details about how the entity manages all payment page scripts that are loaded and executed in the consumer’s browser in accordance with Requirement 6.4.3.</p> <p>Documented evidence that the entity keeps the policies and procedures up to date.</p> <p>Personnel available for interview to confirm that the policies and procedures are:</p> <ul style="list-style-type: none"> ▪ In use. ▪ Known to all affected parties.

Table 6. Testing Procedures and Expected Evidence for Requirement 6.1.2 for Roles and Responsibilities to Support Requirement 6.4.3

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 6.1.2	
Roles and responsibilities for performing activities in Requirement 6 are documented, assigned, and understood.	
Testing Procedure Purpose	Expected Evidence from Entity
6.1.2.a Roles and responsibilities for Requirement 6.4.3 are defined according to Requirement 6.1.2.	Documented definition of roles and responsibilities for performing activities for Requirement 6.4.3.
6.1.2.b Roles and responsibilities for Requirement 6.4.3 are assigned and understood.	<p>Personnel available for interview to confirm that:</p> <ul style="list-style-type: none"> ▪ Roles and responsibilities for performing activities for Requirement 6.4.3 are assigned as documented. ▪ Personnel understand the roles and responsibilities they are assigned related to Requirement 6.4.3.

Table 7. Testing Procedures and Expected Evidence for Requirement 6.4.3

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 6.4.3	
Payment page scripts are managed, including authorization, integrity assurance, and script inventory	
Testing Procedure Purpose	Expected Evidence from Entity
6.4.3.a Policies and procedures are defined.	Policies and procedures that: <ul style="list-style-type: none"> ▪ Define processes for managing payment page scripts. ▪ Include all elements specified in the requirement.
6.4.3.b Personnel confirm that payment page scripts are managed according to the requirement.	Personnel available for interview to confirm that the defined processes and methods are followed for: <ul style="list-style-type: none"> ▪ Authorizing scripts. ▪ Assuring the integrity of scripts. ▪ Maintaining an inventory of scripts, including a technical or business justification.
6.4.3.b Inventory records provide evidence of script inventories and business or technical justification.	Documentation that includes: <ul style="list-style-type: none"> ▪ Script inventory records¹. ▪ Technical or business justification for scripts used.
6.4.3.b System configurations provide evidence of a method to assure script integrity.	System configurations showing the methods for ¹ : <ul style="list-style-type: none"> ▪ Authorizing each script. ▪ Assuring the integrity of each script.

¹ The evidence provided depends on whether manual or automated methods are used.

More information about evidence for Requirement 6.4.3

Examples of authorization records might include, but are not limited to:

- Electronic entries in a workflow system, a service management system, or a ticketing system.
- Retained email approval.

Examples of how script integrity is assured might include, but are not limited to, details about the security controls and techniques implemented to provide this assurance, covered in [Controls and Techniques to Help Meet Requirements 6.4.3 and 11.6.1](#).

Examples of inventory records might include, but are not limited, to a spreadsheet, a document, a database, a page in a content management system, or a tag/script management system.

Testing Procedures and Related Assessment Evidence for Requirement 11.6.1

The tables in this section address the intended purpose of each testing procedure and the evidence that an assessed entity is expected to have available and provide during a PCI DSS assessment of Requirement 11.6.1.

The following tables are included:

- Table 8 summarizes the purpose of the testing procedures and the expected evidence for Requirement 11.1.2, for roles and responsibilities that need to be defined to perform activities in Requirement 11. This table only includes information related to roles and responsibilities for Requirement 11.6.1.
- Table 9 summarizes the purpose of the testing procedures and the expected evidence for Requirement 11.6.1.
- Table 10 summarizes the purpose of the testing procedures and the expected evidence for Requirement 12.10.5. This requirement, for an incident response plan that includes monitoring and responding to alerts, includes a bullet for the change- and tamper-detection mechanism for payment pages. This table only includes information related to incident response processes for monitoring and responding to alerts for Requirement 11.6.1.

Note that there are no policies and procedures identified in Requirement 11.6.1; therefore, this section does not include a table for Requirement 11.1.1.

Table 8. Testing Procedures and Expected Evidence for Requirement 11.1.2 for Roles and Responsibilities to Support Requirement 11.6.1

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 11.1.2	
Roles and responsibilities for performing activities in Requirement 11 are documented, assigned, and understood	
Testing Procedure Purpose	Expected Evidence from Entity
11.1.2.a Roles and responsibilities for Requirement 11.6.1 are defined according to Requirement 11.1.2.	Documented definition of roles and responsibilities for performing activities for Requirement 11.6.1.
11.1.2.b Roles and responsibilities for Requirement 11.6.1 are assigned and understood.	Personnel available for interview to confirm that <ul style="list-style-type: none"> ▪ Roles and responsibilities for performing activities for Requirement 11.6.1 are assigned as documented. ▪ Personnel understand the roles and responsibilities they are assigned related to Requirement 11.6.1.

Table 9. Testing Procedures and Expected Evidence for Requirement 11.6.1

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 11.6.1	
<p>A change- and tamper-detection mechanism is deployed to:</p> <ul style="list-style-type: none"> ▪ Alert personnel to unauthorized modifications to security impacting HTTP headers and script contents of payment pages as received by the consumer browser. ▪ Evaluate received HTTP headers and payment pages. ▪ Perform the mechanism’s functions at least weekly or at the frequency defined in the entity’s targeted risk analysis (TRA) performed according to Requirement 12.3.1. 	
Testing Procedure Purpose	Expected Evidence from Entity
<p>11.6.1.a A change- and tamper-detection mechanism is in use.</p>	<p>Documentation or system configurations that shows:</p> <ul style="list-style-type: none"> ▪ System settings for the change- and tamper-detection mechanism. ▪ Monitored payment pages. ▪ Results from monitoring activities (for example, logs, reports, or other results).
<p>11.6.1.b The mechanism is configured to address all elements specified in the requirement.</p>	<p>System configuration settings showing that the mechanism is configured according to the requirement:</p> <ul style="list-style-type: none"> ▪ It detects and alerts personnel about unauthorized modifications to the security impacting HTTP headers and script contents of payment pages. ▪ It evaluates received HTTP headers and payment pages.
<p>11.6.1.c If the entity documented a TRA to define how frequently the mechanism’s functions are performed, the TRA is completed according to Requirement 12.3.1.</p>	<p>The documented TRA.</p>
<p>11.6.1.d Personnel confirm that the mechanism’s functions are performed at least weekly or at the frequency defined in the entity’s TRA.</p>	<p>Personnel available for interview to confirm that the mechanism’s functions are performed at least weekly or at the frequency documented in the entity’s TRA.</p>
<p>11.6.1.e Configuration settings provide evidence that the mechanism’s functions are performed at least weekly or at the frequency defined in the entity’s TRA.</p>	<p>Configuration settings that show the mechanism’s functions are performed at least weekly or at the frequency documented in the entity’s TRA.</p>

More Information About Evidence for Requirement 11.6.1

Entities can use the script inventory developed to meet Requirement 6.4.3 to configure this mechanism.

Examples of change- and tamper-detection mechanisms include, but are not limited to, automated or manual transaction simulations, real-time monitoring platforms, and third-party script services.

Examples of alerting methods include, but are not limited to, email, SMS/text message, SYSLOG event, Windows security log event, and a vendor log that is part of a proprietary system.

Refer to details about the security controls and techniques that can be used in [Controls and Techniques to Help Meet Requirements 6.4.3 and 11.6.1](#).

Table 10. Testing Procedures and Expected Evidence for the Requirement 12.10.5 for Incident Response Plan Monitoring and Responding to Alerts for Requirement 11.6.1

Testing Procedure Purpose	Expected Evidence from Entity
Requirement 12.10.5	
<p>The security incident response plan includes monitoring and responding to alerts from security monitoring systems, including but not limited to the change- and tamper-detection mechanism for payment pages.</p> <p><i>Note that other bullets in Requirement 12.10.5 not directly relevant to payment pages are not included in this table.</i></p>	
Testing Procedure Purpose	Expected Evidence from Entity
12.10.5.a The documented incident response plan includes monitoring and responding to alerts from the change- and tamper-detection mechanism for payment pages.	The documented incident response plan that details the monitoring and response processes for alerts from the change- and tamper-detection mechanism for payment pages.
12.10.5.b Incident response processes include monitoring and responding to alerts from the change- and tamper-detection mechanism for payment pages.	Opportunity to observe incident response processes. <i>Note that it may be necessary to observe incident response “table-top exercises” if there are no actual incidents for observation.</i>

Further Resources and Guidance

The following resources and guidance documents are available from the Document Library on the PCI Security Standards Council website.

- Information Supplement: Best Practices for Securing E-commerce
- SAQ Instructions and Guidelines
- Small merchant documents

About the PCI Security Standards Council

The PCI Security Standards Council (PCI SSC) is a global forum for the ongoing development, enhancement, storage, dissemination, and implementation of security standards for account data protection. Our role is to enhance global payment account data security by developing standards and supporting services that drive education, awareness, and effective implementation by stakeholders. To learn more, please visit pcisecuritystandards.org.