PCI DSS V4 and E-Commerce

JavaScript Integrity Requirements

by John Elliott, Security Advisor

Table of contents

01: Executive Summary

PCI DSS version 4 contains two new requirements aimed at ensuring the integrity of pages where payment is taken on an e-commerce website. The first requirement (6.4.3) is designed to minimize the attack surface and manage all JavaScript present in the **Payment Page**[1]. The second requirement (11.6.1) aims to detect tampering or unauthorized changes to the payment page, and generate an alert when such changes are detected.

In addition to applying to the payment page, these requirements also apply to a Parent Page hosting any iframe elements (or the URI for a redirection) for merchants that-would have typically have met the eligibility criteria of SAQ A.

PCI DSS v4 is mandatory for assessments after 31st March, 2024. These two new requirements are "a best practice until 31 March 2025" meaning that they will only be tested in assessments after 31st March 2025.

SAQ A Eligible Merchants will need to be able to meet these new requirements for the parent page that contains the redirection URI or loads JavaScript from the Payment Service Provider[2] (PSP) which creates the iframe or individual hosted fields in iframes.

SAQ A-EP, all other e-commerce merchants and PSPs will need to be able to meetthese new requirements for their payment page.

Jscrambler's Webpage Integrity can help Merchants and PSPs to meet these challenges. This white paper assumes a familiarity with PCI DSS along with the use and eligibility criteria of the PCI DSS Self-assessment Questionnaires (SAQs).

1 This is now a defined term in PCI DSS v4

2 Includes payment gateways, acquirers etc, any entity that provides a way of collecting payment card data from a consumer and submitting to an acquirer / card brand network for authorization.

02: PCI DSS Version 4

Timetable

	Date
V4.0 released:	31 March 2022
Mandatory for assessments after:	31 March 2022
New 'future dated' requirements (which includes the two JavaScript integrity requirements) to be included in assessments after:	31 March 2022

Validation Options in PCI DSS v4

In PCI DSS v3 and earlier, there was only one way of meeting a requirement – a prescriptive requirement was defined along with a prescriptive testing procedure.

In PCI DSS v4, there are two ways of meeting a requirement, via the **Defined Approach** – which is the new term, for the traditional prescriptive approach of PCI DSS – and the **Customized Approach** which allows an entity to meet as security objective using their own selection of controls. The security objective is called the **Customized Approach Objective**. The Customized Approach can only be used by entities undergoing an ISA or QSA assessment (i.e. not an SAQ).

Requirement 6.4.3 (Preventative)

This requirement is designed to make sure that all JavaScript included in the payment page is actively managed. The aim is to ensure that the attack surface is minimized by requiring an approval process for each script added to the payment page, and that a way of validating the integrity of a script is defined to ensure that malicious scripts are not placed on the payment page.

Defined Approach

“All payment page scripts that are loaded and executed in the consumer’s browser are 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 justification as to why each is necessary.”

Customized Approach Objective

“Unauthorized code cannot be present in the payment page as it is rendered in the consumer’s browser.”

Requirement 11.6.1 (Detective)

This requirement aims to detect and alert on unauthorized changes to the payment page indicative of a skimming-type attack. There is no requirement to block, just to alert.

Defined Approach

“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 HTTP headers and the contents of payment pages as received by the consumer browser.
- The mechanism is configured to evaluate the received HTTP header and payment page.

The mechanism functions are performed as follows:

- At least once every seven days.

  or

- Periodically (at the frequency defined in the entity's targeted risk analysis, which is performed according to all elements specified in Requirement 12.3.1)."

Customized Approach Objective[3]

"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."

3 This customized approach objective is rare in the standard in having a two-part objective

Defined Approach Summary

Taking the above requirements, Jscrambler's analysis of what an entity would need to do to be able to be assessed to fully meet the requirement are:

Features:	Notes:
Requirement 6.4.3	
Records that each script is authorized	Stated requirement
Records who authorized the script	Although management approval is indicated in the guidance, it could be authorized by anyone or any process
Records when a script was authorized	Not explicitly in the standard but advisable to demonstrate to an assessor that the authorisation wasn't just 3 minutes before the assessor arrived. I.e. the use of the script and its authorisation were reasonably contemporaneous
Script is listed in an inventory	Stated requirement
Records when a script was first deployed to the payment page	Necessary for the assessor to determine the accuracy of the inventory

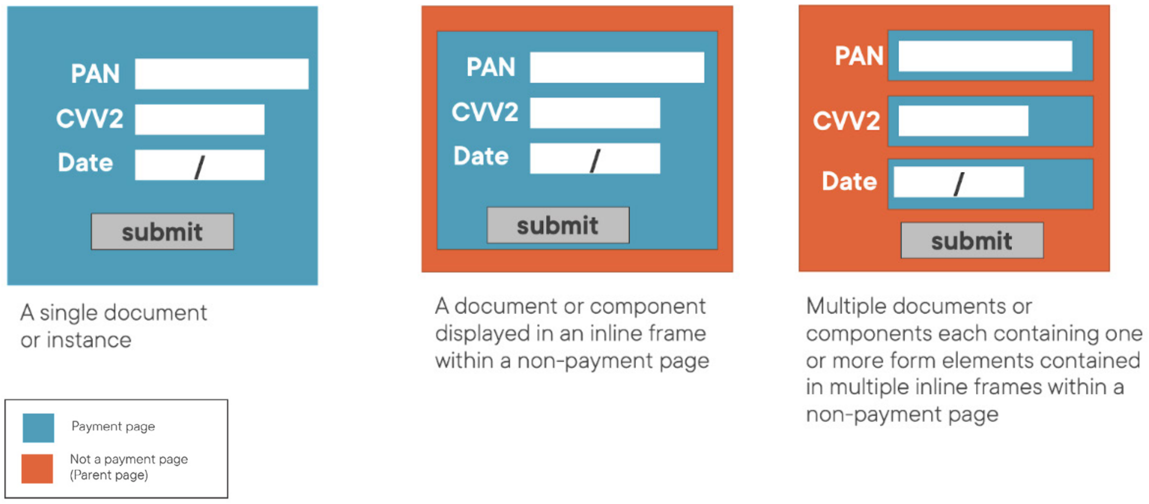
Requirement 6.4.3	
Records when a script was removed from the payment page	Necessary for the assessor to determine the accuracy of the inventory
Records the technical analysis of functionality of each script	Necessary to document the actual functionality of a script so that the justification is attached to the functionality (a single script can include multiple functionalities)
Records justification of necessity for all functionality contained in each script	Stated requirement – must be “written” and the justification is for each functionality
Validates the integrity each of the script	Stated requirement
Maintains a log of how integrity was validated	There are multiple ways of validating integrity, some could be manual, some could be enforced by technology, however a log should be maintained to show that for each script, the integrity mechanism was operational. A competent assessor would look for this

Features:	Notes:
Requirement 11.6.1	
Validation mechanism checks HTTP Header	Stated requirement
Validation Mechanism checks contents of payment page	Stated requirement
Validation mechanism alerts on unauthorized modification of HTTP Header	Stated requirement
Validation mechanism alerts on unauthorized modification of contents of payment page	Stated requirement
Validation mechanism provides a log function	So that the assessor can validate the periodicity of operation and match it either with the stated requirement (< = 7 days) or the period determined by the TRA

Definitions and SAQ A

The two new requirements apply to the **Payment Page**.
There is a formal definition of the payment page which is in the glossary in Appendix G of PCI DSS v4.

A web-based user interface containing one or more form elements intended to capture account data. The payment page can be rendered as any one of:


PAN
CVV2
Date
/
submit
A single document or instance
PAN
CVV2
Date
/
submit
A document or component displayed in an inline frame within a non-payment page
PAN
CVV2
Date
/
submit
Multiple documents or components each containing one or more form elements contained in multiple inline frames within a non-payment page
Payment page
Not a payment page (Parent page)

The original intention of the requirement is it applies to the blue page elements that contain the form field elements used to capture cardholder data (pages, iframes) and not the orange **Parent Page** because of the separation of an iframe due to the same origin policy – i.e. malicious JavaScript running in the parent page is not able to access the cardholder data as it is entered into the payment page.

SAQ A

The large majority of e-commerce merchants use an iframe / hosted field solution provided by their payment service provider/processor (PSP). The integration of the iframe into the merchant's checkout page is usually via a JavaScript bundle provided by, and loaded from, the PSP allowing them to meet the eligibility criteria of SAQ A. There are, however, known successful skimming attacks against an iframe (Frame overlay, AiTM/AiTB etc) and it is assumed that for this reason that the PCI SSC has extended the scope of these two new requirements in SAQ A (where the merchant has no payment page), with the following guidance:

6.4.3

For SAQ A, Requirement 6.4.3 applies to the page(s) on the merchant's website(s) that provides the address (the URL) of the TPSP's payment page/form to the merchant's customers.

11.6.1

For SAQ A, Requirement 11.6.1 applies to merchants that include a TPSP's inline frame (iframe) payment form on the merchant's website.

SAQ A-EP

SAQ A-EP is appropriate when the merchant's website is in control of the creation of the payment form - which typically now would be by just including the PSP's JavaScript bundle - BUT the payment fields are not in an iframe and so any JavaScript included in the page would be able to read and copy any elements from the DOM. This architecture was previously known as a "Direct Post" - the merchant's website did not store, process, or transmit cardholder data but if compromised would allow a skimming attack. This is the attack that the requirement is designed to defeat.

Both the two new requirements are included in SAQ A-EP.

03: Considerations for Merchants

The compliance consideration for a merchant differs depending on the solution provided: whether that is via hosted fields / iframes or via a direct post / JavaScript form that provides no isolation between the merchant's website and the payment fields.

Methods of Payment Acceptance

A number of ways of accepting payments have been designed over the past few years, and the way these are structured affects the compliance requirements of the merchant. They fall into the following categories.

The two entities involved in the process are:

Merchant

The retailer – in many cases they will also be a merchant in the payment card sense of the word with responsibilities to maintain PCI DSS compliance themselves and have a Merchant Identifier (MID) issued by an acquirer or card brand. They may also just be a customer of a payment facilitator who may have also subsumed the merchant's actual PCI DSS compliance responsibilities.

PSP

A Payment Service Provider / Payment Processor / Payment Gateway or payment facilitator.

Type and Description:	**Who generates the payment page?**	**Where does the browser send the data?**	**Who generates the parent page?**
Redirection	PSP	PSP	Merchant
IFRAME or hosted fields in individual IFRAMEs within a merchant's PARENT PAGE	PSP	PSP	Merchant (may call JavaScript from the PSP)
Direct post - merchant's own code creates form fields or JavaScript Form calls JavaScript from the PSP. Form created in the page, not an IFRAME	Merchant (may call JavaScript from the PSP)	PSP	-
API - out of scope of this paper	Merchant	Merchant	-

Typical PCI DSS Validation

This table shows the typical PCI DSS validation required for the merchant. The designation RoCSAQ X is intended for Level 1 or 2 merchants who may be required to have a QSA or ISA assessment but who meet the eligibility criteria for an SAQ and so just test the requirements in the relevant SAQ following PCI SSC FAQ 1331.

Type and Description:	Who generates the parent page?
Redirection	SAQ A or RoC$_{\text{SAQ A}}$
IFRAME or hosted fields in individual IFRAMEs within a merchant's PARENT PAGE	SAQ A or RoC$_{\text{SAQ A}}$
Direct post – merchant's own code creates form fields or JavaScript Form calls JavaScript from the PSP. Form created in the page, not an IFRAME	SAQ A-EP or RoC$_{\text{SAQ A-EP}}$
API - out of scope of this paper	SAQ D or RoC

Meeting the New Requirements

The new requirements have been written in a technologically neutral way and so it is anticipated that they can be fulfilled in a number of ways. The following examples are offered in the standard.

CSP and SRI

Together setting a content security policy (CSP) defining the locations that scripts can be loaded from (and data sent) and using SRI to validate the integrity of all JavaScript loaded will meet the majority of the requirements. The language used in the requirements suggests that this is what the SSC had in mind in drafting.

Tag or script management

Tag or script management can provide workflows for the authorization and inventory of scripts. More advanced solutions would be able to provide automated inventory along with an analysis of the functionality and risk associated with each script.

External monitoring / scanning

Scanning or synthetic user monitoring is able to build inventories of scripts which could include approval workflows. Scanning would also be able to detect and alert on changes to the scripts present on a page and provide analysis of those likely to be malicious.

Tamper detection and tamper resistance

Sandboxing of scripts would be able to provide alerts on both changes to script behavior but also indicators of malicious activity.

Reverse proxies and CDNs

The use of a reverse proxy or integrity checks within a CDN would allow for unauthorized changes to scripts to be detected and could include functionality for maintaining an inventory and approvals.

SAQ A Solutions

SAQ A solutions used by merchants are iframes, hosted fields (in iframes), and redirection. Because of the extension of the new requirements to the Parent Page, both the merchant and the PSP will each have responsibilities.

	Parent Page	**Payment Page**
Authorize scripts	Merchant	PSP
Assure integrity	Merchant	PSP
Maintain inventory	Merchant	PSP
Monitor and alert of changes to page contents (New / changed scripts)	Merchant	PSP
Monitor and alert of changes to HTTP headers	Merchant	PSP

Merchant's responsibility for the Parent Page

In many cases, SAQ A merchants have been shielded from the complexities of PCI DSS by PSPs moving to a largely iframe / hosted fields model.
A merchant will now need to fulfill the requirements in respect of all JavaScript on the parent page, which includes the JavaScript provided by the PSP.

Technical approach taken by merchant to meet the new requirements	**Potential Issues**
Content Security Policy	Changes to host locations by third party providers and PSPs may break the page functionality. Merchants will need to have a process to respond to alerts
Subresource integrity	When the PSP updates their JavaScript bundle, the merchant will need to update the HTML in the parent page to include the new SRI hash. If the merchant fails to do this, the PSP's JavaScript will silently not be loaded and customers will not be able to complete their purchases. As PSPs frequently update their JavaScript bundles used to create the iframe or hosted fields, this has the potential to break many sites
Tag / Script Manager	Merchants will need procedures in place to respond to alerts and verify that new scripts and changes to existing scripts were authorized. Different solutions may provide intelligence (e.g. "this script contains similar code to a frame-overlay attack") to assist merchants responding to alerts
Monitoring / Scanning	
Tamper / Change Detection	
Reverse Proxies	

SAQ A-EP Solutions

SAQ A-EP solutions offered by PSPs tend to start with a JavaScript bundle that created the payment form, with the submit then posted to an API at the PSP. Here there is no Parent Page, just the Payment page.

	Payment Page
Authorize scripts	Merchant
Assure integrity	Merchant
Maintain inventory	Merchant
Monitor and alert of changes to page contents (New / changed scripts)	Merchant
Monitor and alert of changes to HTTP headers	Merchant

Merchant's responsibilities for the payment page

The merchant needs to both manage the PSP JavaScript and all other "necessary" JavaScript on the payment page.

Technical approach taken by merchant to meet the new requirements	Potential Issues
Content Security Policy	Depending on how much third-party JavaScript is included in the payment page, and how frequently that JavaScript may change, using CSP and SRI may be appropriate. Again, if the merchant loads a JavaScript bundle from the PSP to create the payment form then any changes by the PSP to the destination data is posted or the JavaScript bundle itself will stop payments being made Merchant
Subresource integrity	
Tag / Script Manager	Same issues as SAQ A described above
Monitoring / Scanning	
Tamper / Change Detection	
Reverse Proxies	

API Solutions

Merchants using an API will create the payment page, receive the cardholder data back to the merchant's infrastructure, and then process the payment by sending the cardholder data to a PSP.

The merchant typically does not use PSP-originated JavaScript to create the payment form, so will just need to follow the new requirements in respect of all other JavaScript they may load into the payment page.

	Payment Page
Authorize scripts	Merchant
Assure integrity	Merchant
Maintain inventory	Merchant
Monitor and alert of changes to page contents (New / changed scripts)	Merchant
Monitor and alert of changes to HTTP headers	Merchant

Merchant's responsibilities for the payment page

The merchant needs to both manage the PSP JavaScript and all other "necessary" JavaScript on the payment page.

Technical approach	Potential Issues
Content Security Policy	There are fewer issues with the API model as the merchant is in full control of any JavaScript used to create a payment form and so can isolate the functionality of payments from other JavaScript on the page. The merchant can incorporate making necessary changes to the CSP and SRI as part of the change/release control process. So there is less risk in a change to payment-form- related JavaScript breaking the functionality of the site. However, all other non-payment- related JavaScript that the merchant deems "necessary" will need to be managed and alerts responded to when changes occur.
Subresource integrity	
Tag / Script Manager	
Monitoring / Scanning	
Tamper / Change Detection	
Reverse Proxies	

04: Jscrambler Webpage Integrity

Jscrambler Webpage Integrity (WPI) is a holistic solution to detect and block, in real-time, malicious behavior on the client-side of web applications. It prevents leaking or scraping of sensitive data and protects against web supply chain attacks like Magecart – WPI addresses both of the new requirements.

Webpage Inventory

Complete visibility of every script and network request on your website. Simplifies the identification of malicious client-side behavior and vetting of resources.

Third-party management

Simple onboarding and vetting of third-party scripts, with full observability of each script and a powerful rules engine that allows controlling their behaviors.

User data management

Dashboard with details of how user data is being handled on the client-side, containing data leakage insights. Provides control over this data, preventing leakage attempts.

Webpage threat mitigation

Powerful and granular rules engine that blocks any script in real-time if it exhibits malicious or disallowed behavior (e.g. formjacking, DOM tampering, skimming, data leakage).

Jscrambler Webpage Integrity is completely plug-and-play and does not require installing anything locally, nor any input from the end-user. This module is protected by Jscrambler Code Integrity, preventing any sort of bypass or tampering.

05: Using Jscrambler to meet the new requirements

Merchants that load JavaScript from a PSP to create the payment functionality

For SAQ A and SAQ A-EP eligible merchants that:

1. Serve a page containing the URI for redirection (SAQ A), or
2. Load a JavaScript bundle from the PSP to:
    - Create an iframe of a number of hosted fields in an iframe (SAQ A), **or**
    - Create a payment form that when completed, the cardholder data is transmitted directly to the PSP (SAQ A-EP).

Jscrambler's Webpage Integrity provides full compliance with the new DSS requirements and is the easiest way to comply without the risk of breaking the functionality of the page.

Merchants not reliant on PSP JavaScript

Webpage Integrity provides complete JavaScript integrity management for the payment page and so is extremely beneficial as the payment page is likely to include other JavaScript libraries/bundles loaded from third parties WPI would provide a much smoother and more practically manageable solution than using CSP and SRI and allow the merchant to be fully compliant with the new requirements 6.4.3 and 11.6.1 in PCI DSS 4.0.

If you want to know more about how Jscrambler can help you prevent client-side attacks, don't hesitate to contact us.

hello@jscrambler.com | +1 650 999 0010