6.4.3 & 11.6.1:
Do You Understand Website Scoping?

Special thanks to Steven Eric Fisher & Carla Brinker

Jeff Man

Qualified Security Assessor
Sr. Information Security Consultant

online business systems

Jeff Zitomer

Sr. Director of Product Management,
Client-Side Defense & PCI DSS

HUMAN

PCI Security Standards Council

Agenda

1. A quick recap on 6.4.3 and 11.6.1
2. How browser mechanics and Web application architecture impact application segmentation, the client-side attack surface, and PCI DSS scope
3. Properly segmenting web applications to reduce client-side scope
4. Actionable next steps for merchants, TPSPs, and QSAs

What Are the New Browser Script Requirements?

6.4.3
Script Management

- Maintain inventory with written justification
- Confirm scripts are authorized
- Assure scripts' integrity

11.6.1
Change and Tamper Detection

- Alert to unauthorized modification to security-impacting HTTP headers and the script contents of payment pages, as received by the consumer browser

*Reference full requirements in Payment Card Industry Data Security Standard: Requirements and Testing Procedures, v4.0.1

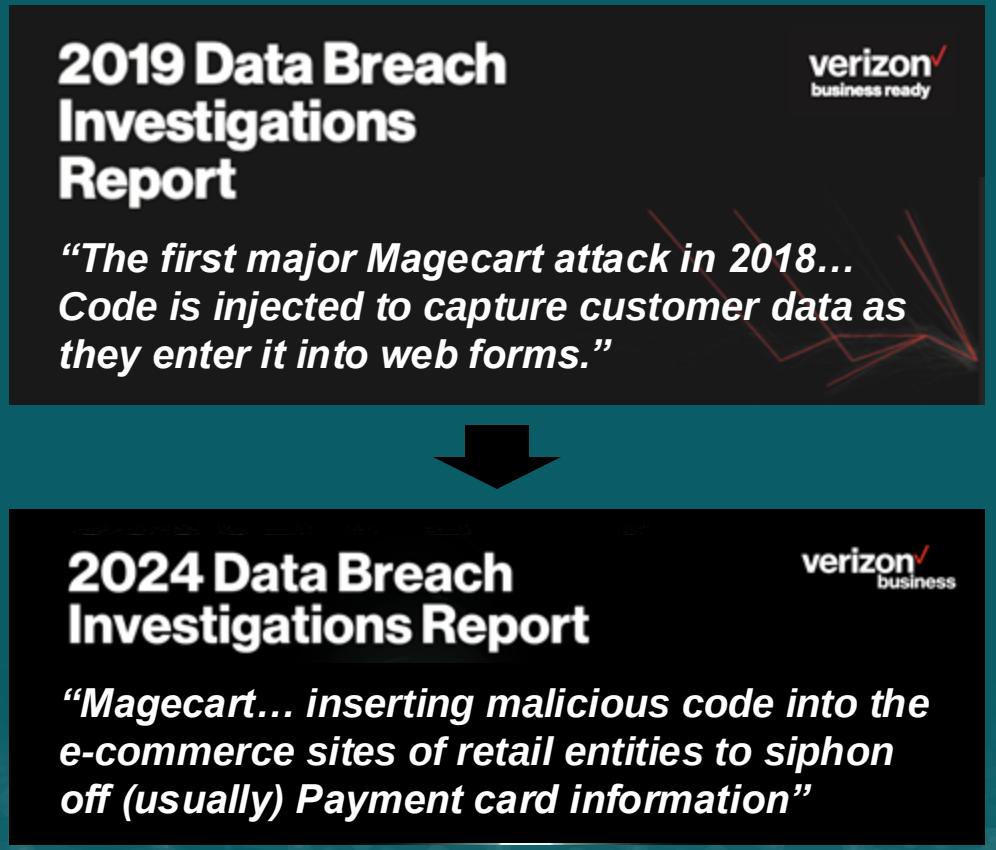
PCI Security Standards Council

Why Secure Scripts?

A lucrative, vulnerable, and widely exploited attack surface

Scripts are

- Plentiful
- Business-critical
- Bypass change management and security controls
- Access cardholder data freely



PCI Security Standards Council

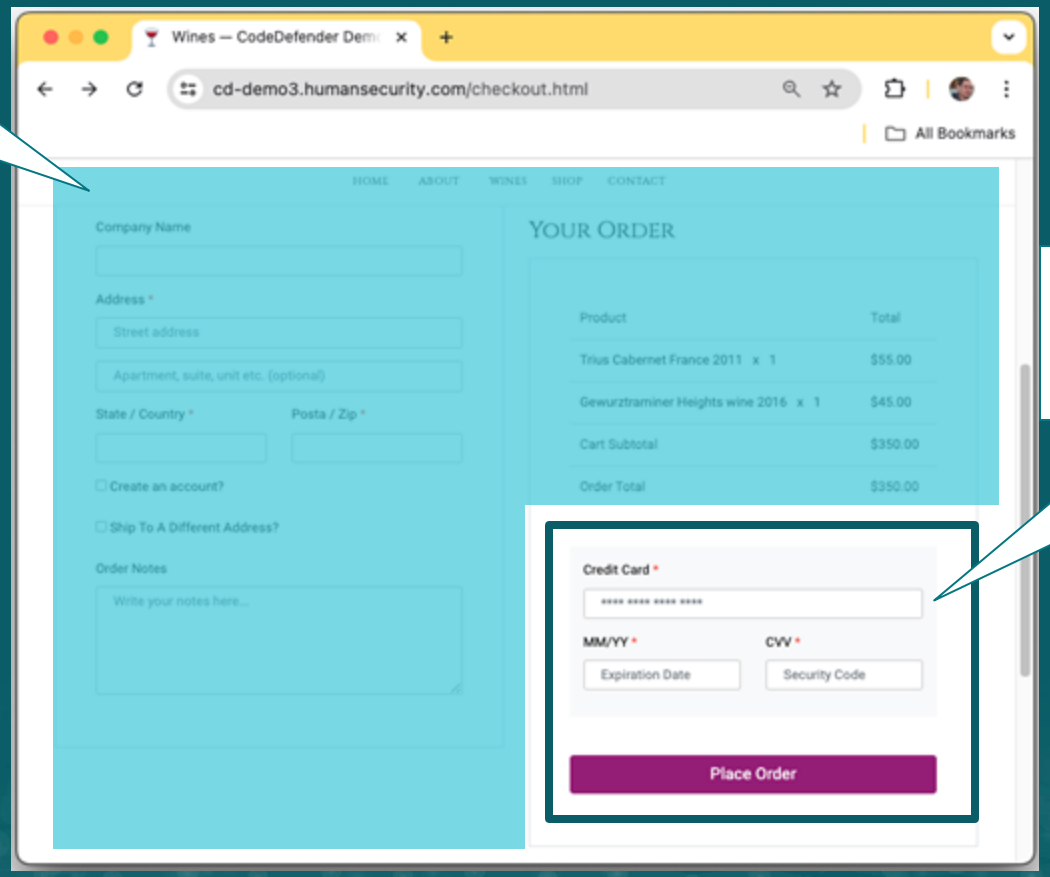
Now Let's Dwell on That Point for a Moment…

Trust is a dicey subject,
everyone wants to be trusted
but only a few people are willing
to put in the work to show themselves trustworthy.

4.0.1 Clarifies Merchant vs. TPSP Ownership, but What Are the Merchant's Scoping Boundaries?

Is this view the entity's "webpage(s)"?

*"...applies to **scripts in the entity's webpage(s)** that includes a payment processor's embedded payment page/form"*



Is the embedded form adequately segregated?

"Scripts in the embedded payment page/form are the responsibility of the payment processor"

PCI Security Standards Council

Source: Payment Card Industry Data Security Standard: Requirements and Testing Procedures, v4.0.1

First, Let's Look Underneath the Browser's Hood

PCI Security Standards Council

The Browser Loads Resources Into “Sandboxes”

Sandboxes enable application code to **execute freely** within a **restricted environment**

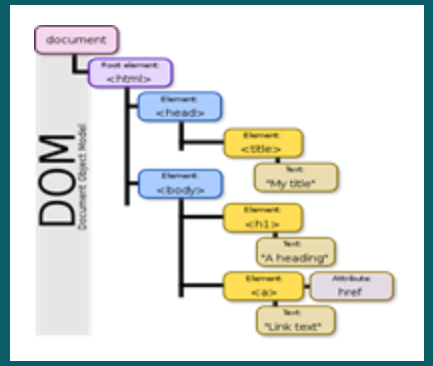
Sandbox boundaries *generally* map to tabs, host domains, webpages, or iframes

PCI Security Standards Council

Within Each Sandbox, Javascript Can…

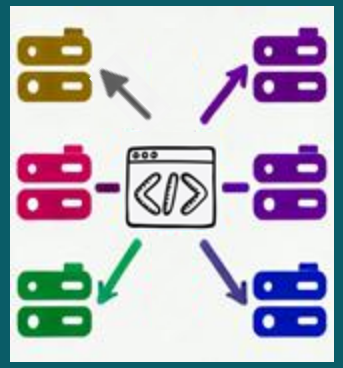
Access/modify the DOM*

- Read cardholder data
- Present fake payment form
- Inject inline scripts



Communicate with domains

- Load from malicious.com
- Steal PII & cardholder data
- Redirect to fake payment form



Access/modify local storage

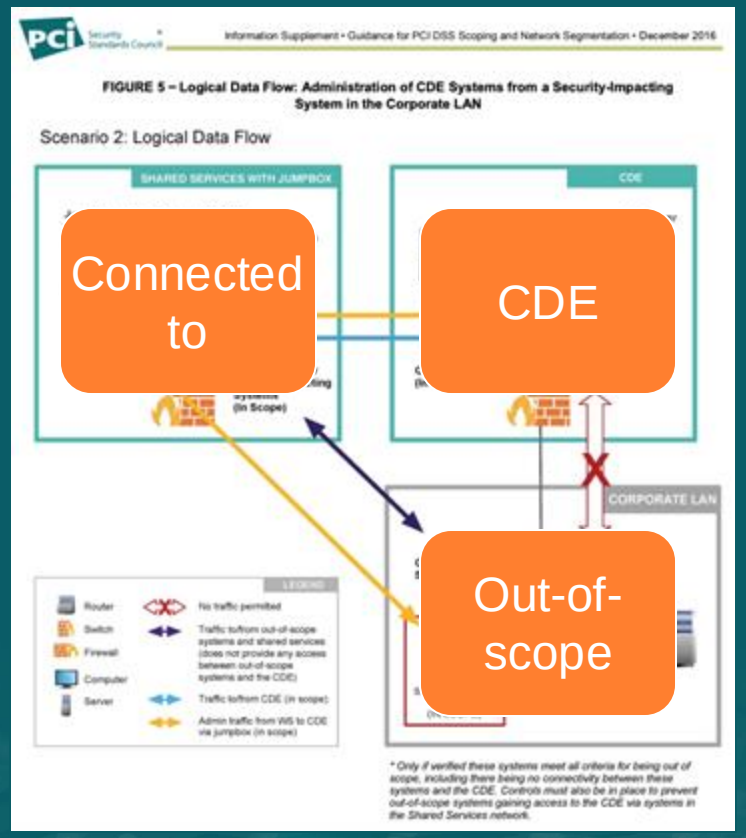
- Session hijacking
- Read/modify sensitive data

The browser does not natively segment within a sandbox

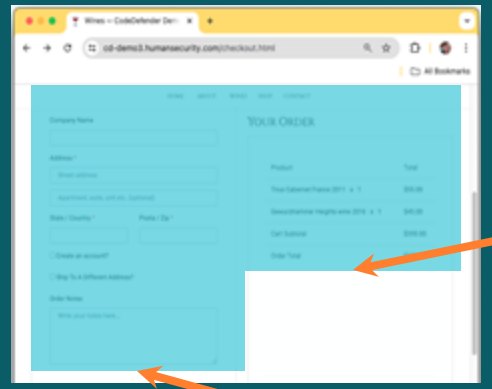
PCI Security Standards Council

*DOM: Document Object Model - organizes all of the page elements

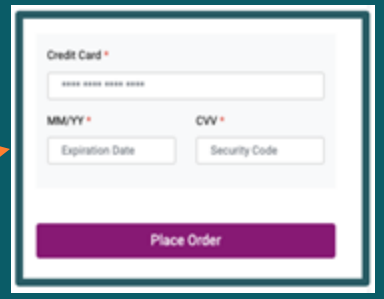
Borrowing Concepts from PCI Lore, Let's Examine the Implied Webpage Scoping Boundaries



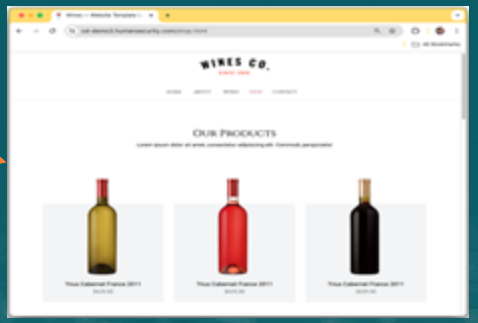
"Checkout page"



1 Connected-to vs. CDE



2 Connected-to vs. Out-of-scope



PCI Security Standards Council®

Source: Information Supplement - Guidance for PCI DSS Scoping and Network Segmentation - December 2016

1

Connected-to vs. CDE: Set the “Sandbox” Attribute and VERY Carefully Lift Restrictions

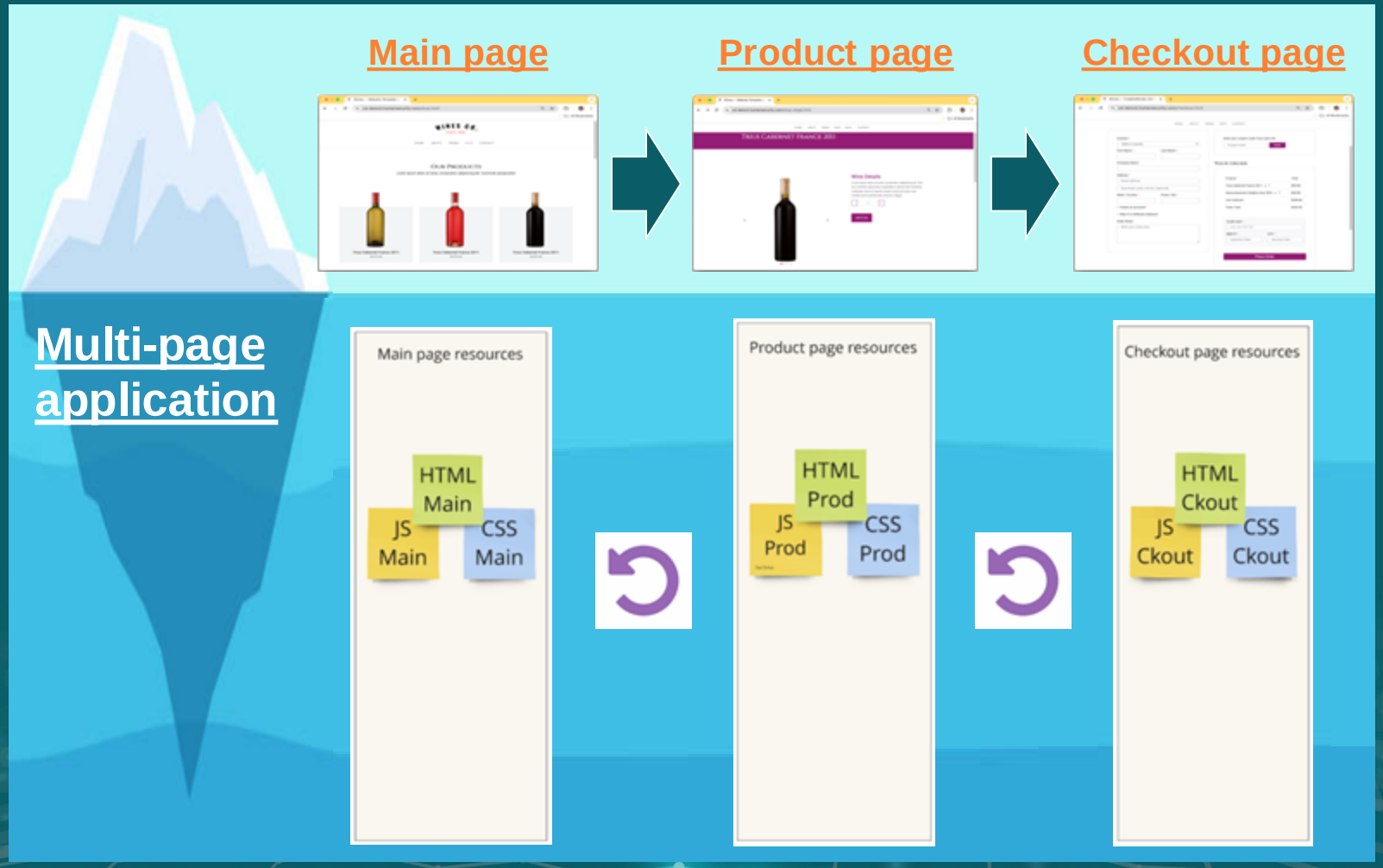
- Always use <iframe src=... **sandbox**></iframe>
- Apply the “least privilege” concepts when lifting restriction (e.g., **allow-scripts** may be necessary)
- **Avoid**
  - **allow-same-origin**
  - **allow-top-navigation**
  - **allow-popups-to-escape-sandbox**

* iframe creation method (html tag vs. TPTP script) may impact eligibility for SAQ A vs. SAQ A-EP
** **”Security-impacting headers”** *should* be layered on-top, but deserve a full talk of their own

PCI Security Standards Council

2

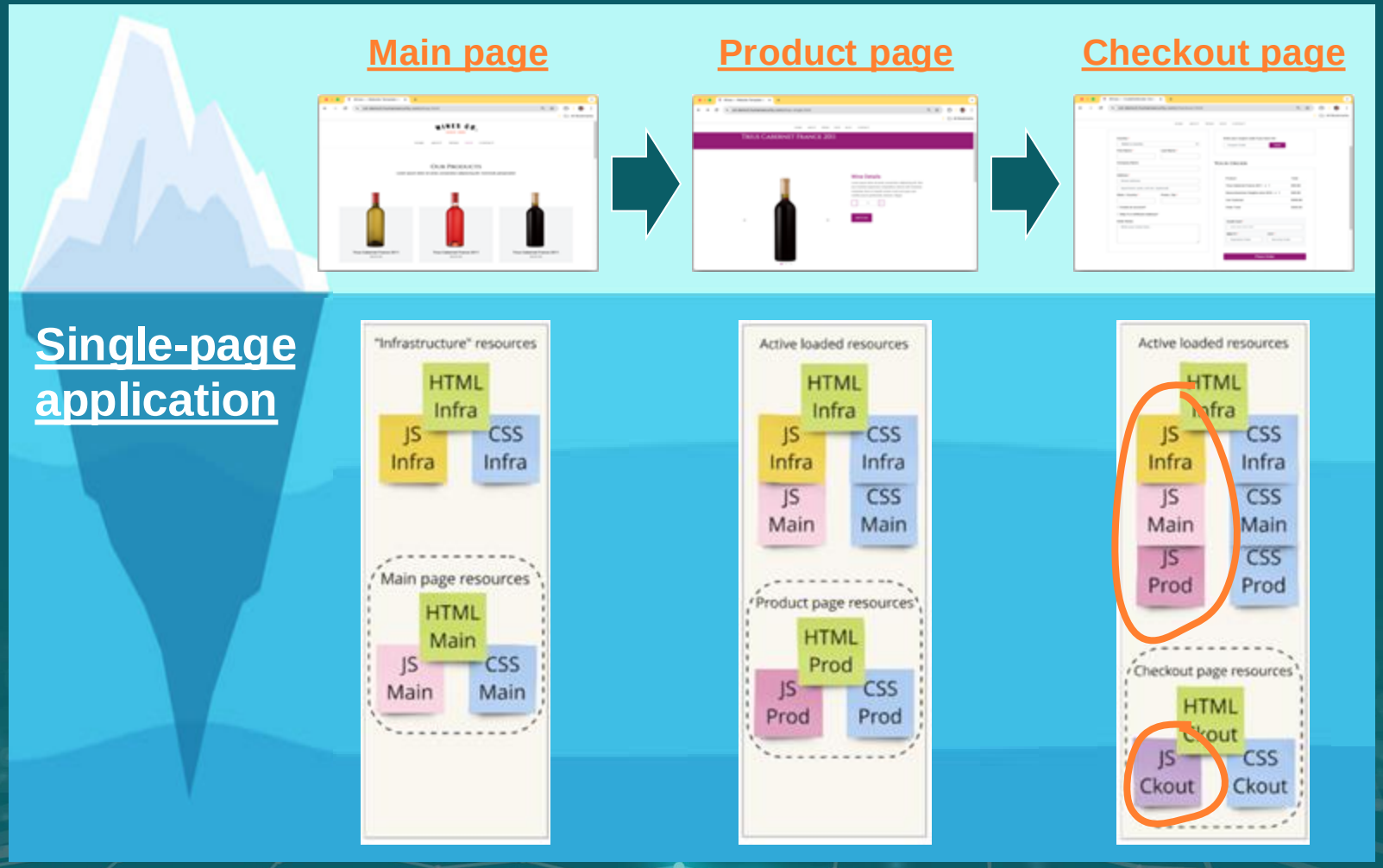
Connected-to vs. Out-of-Scope: In Traditional Web Apps, the Browser Natively Sandboxes Visible Pages



- A URL change initiates the navigation
- Every page loads “fresh” and includes only its own resources and **security-impacting headers**
- **The Connected-to script inventory is “tight”**

2

Connected-to vs. Out-of-Scope: In Contrast, an Entire Modern Single Page App Shares One Sandbox



- Code replaces page content without actually navigating
- Code "cosmetically" changes the url
- **Security-impacting headers are loaded once, on the main page, so are overly-broad**
- **The Connected-to script inventory includes all scripts loaded along entire journey**

To Summarize the Scoping Boundary Topic

Connected-to

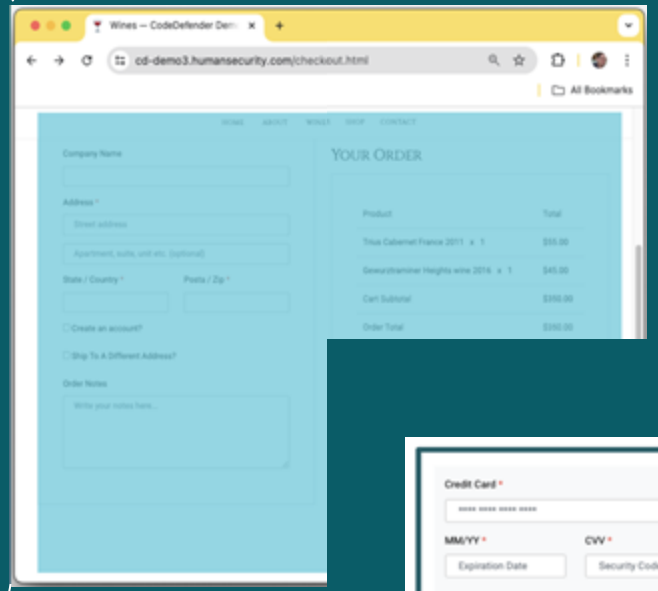
Is “/checkout” truly a separate page?

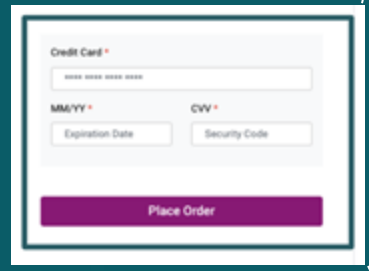
Yes

“/checkout”

No

The entire app!





The CDE

Is the iframe properly implemented?

Yes

✓

PCI Security Standards Council

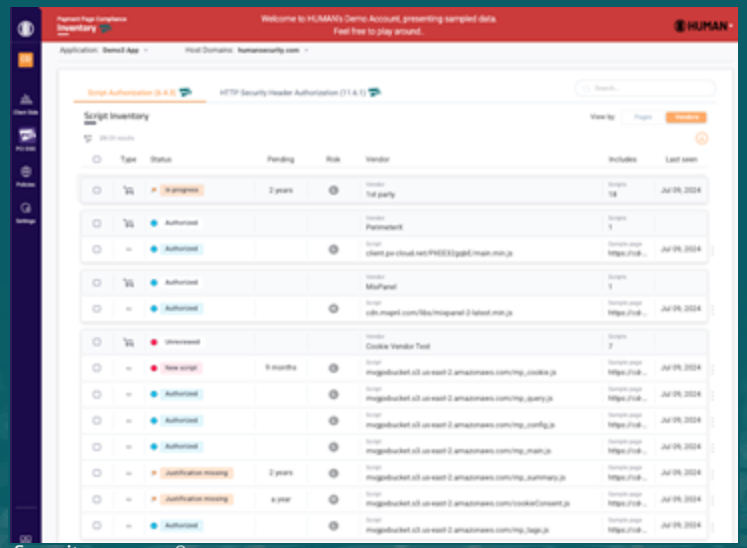
So How Do You Scope a Modern Single Page App?

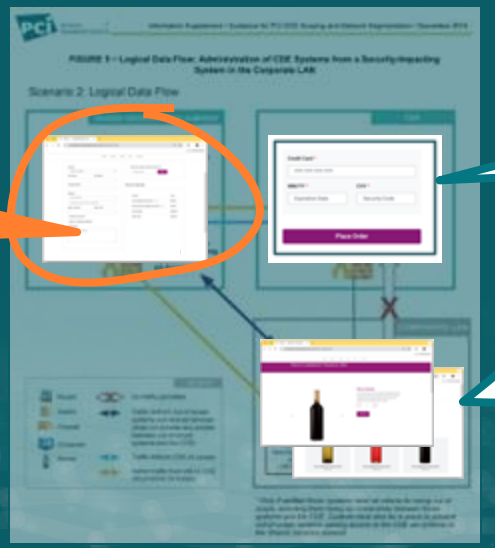
Option 1: Scope the entire SPA

- Massive attack surface
- Broad security-impacting http headers
- Are all SPA scripts “necessary” & compliant?

Option 2: Descope by segmenting the SPA

- Create a “Connected-to intermediary” page that fully loads separately from the SPA





PCI Security Standards Council

The “Connected-to Intermediary Page” Has Tradeoffs

Pros

- **Creates a narrow, and robust surface area (scope) to secure**
- **Drastically reduces ongoing effort** to manage scripts and headers per PCI DSS compliance

Cons

- **Large re-architecting effort!**
- Harder to create a seamless user experience (transfer context)
- Performance

Next Steps

Know the Web application

1. Is the payment service provider's iframe created by script or by html tag? Was it created with the right attributes? Does SAQ A-EP or SAQ A apply?
2. Is the site a single page application? If so, consider creating a "Connected-to" intermediary page
3. Which scripts are in-scope and essential for payment processing?
4. Which vendors are involved (have to be PCI validated and compliant third parties to use SAQ A – including whoever develops your website)

As an Assessor

- SAQ A can be used if
  - The merchant's website includes a redirection method such as a URL redirect or a payment form such as an iframe
  - All elements of the payment page(s)/form(s) delivered to the customer's browser originate only and directly from a PCI DSS compliant TPSP/payment processor.
- SAQ A-EP is used when the merchant is responsible for some or all elements of the payment page and then passes data to a payment service provider
- If JavaScript creates the iframe or the merchant's application code controls how the user is redirected to the payment processor, SAQ A-EP may apply

As a Merchant Prepping for an Assessment

- Scope and document your web app architecture
  - Single page app?
  - Method of redirect (e.g., iframe creation method and attributes)
  - All domains that contain payment-accepting web applications (same origin)
  - List of all elements that load on the payment page (including source)
  - Where and how sandboxing is used
  - Other security features like SRI and CSP in use

Thank you!

A huge thank you to Eric Fisher. Without him, this presentation would not have been possible.

PCI Security Standards Council®


PCi
Security
Standards Council
®