Same-Origin Policy: Evaluation in Modern Browsers

Jörg Schwenk, Marcus Niemietz, Christian Mainka

Ruhr-University Bochum
Contents

1. Introduction & Foundations
2. Methodology & Evaluation
3. Limitations & Access Control Policies
4. Conclusions & Future Work
1. Introduction & Foundations
Same-Origin Policy

https://bank.com

e.g., main HTML document

doctype

HTML 5

document

<html>

head

<head>

<body

<body>

<script src="URL1">

<img src="URL3" name="bear">

img.src=URL3

<link src="URL4">

window

Same-Origin Policy

https://bank.com
### Same-Origin Policy

<table>
<thead>
<tr>
<th><a href="https://bank.com">https://bank.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>IBAN: DE 2345 7568 4013</td>
</tr>
<tr>
<td>Amount: $50</td>
</tr>
</tbody>
</table>
Same-Origin Policy

https://bank.com

IBAN: DE 2345 7568 4013
Amount: $10

JavaScript
Same-Origin Policy

https://attackers.org

https://bank.com

IBAN: DE 1337 0000 0000
Amount: $10,000

JavaScript
Same-Origin Policy

https://attackers.org

https://bank.com

IBAN: DE 2345 7568 4013
Amount: $50

JavaScript
DOM-SOP

e.g., main HTML document

doctype
HTML 5
document
<html>

head
<head>
<body
<body>

<iframe src="URL2"
    id="ID1">
</iframe>

window

window.
frames[0]
id=ID1

document
<html>
doctype
XHTML

head
<head>
<body
<body>
Different Subsets Of SOP Rules

• DOM access (SOP-DOM)
• Local storage and session storage
• XMLHttpRequest
• Pseudoprotocols
• Plugins (e.g., Flash, Silverlight, PDF)
• Window/tab
• HTTP cookies
Focus

• Subset of SOP rules according to these criteria
  – Browser Interactions
    • Interaction of web objects once they have been loaded
  – Web Origins (RFC 6454 as a foundation)
    • “An image is passive content and, therefore, carries no authority, meaning the image has no access to the objects and resources available to its origin”
Scalable Vector Graphics

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="300">
  <script>alert(1)</script>
  <circle cx="120" cy="120" r="110" fill="#fff" stroke="#000" stroke-width="8"/>
</svg>
```

```html
<img src="test.svg">
<embed src="test.svg">
```
Research Questions

• How is SOP for DOM access (SOP-DOM) implemented in modern browsers?
• Which parts of the HTML markup influences SOP-DOM?
• How does the detected behavior match known access control policies?
2. Methodology & Evaluation
SOP-DOM Setup: Test Cases
Your-SOP.com Testbed

ED: JPG and PNG

- EE: `<img>`
- EE: `<canvas>`

ED: Scalable Vector Graphics (SVG)

- EE: `<img>` and `<canvas>`
- EE: `<iframe>` `<object>` and `<embed>`

<table>
<thead>
<tr>
<th>FROM</th>
<th>EE</th>
<th>TO</th>
<th>r</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td><code>&lt;iframe&gt;</code></td>
<td>ED</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>HD</td>
<td><code>&lt;object&gt;</code></td>
<td>ED</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>HD</td>
<td><code>&lt;embed&gt;</code></td>
<td>ED</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>HD</td>
<td><code>&lt;iframe&gt;</code></td>
<td>ED</td>
<td>no*</td>
<td>no*</td>
</tr>
<tr>
<td>HD</td>
<td><code>&lt;object&gt;</code></td>
<td>ED</td>
<td>no*</td>
<td>no*</td>
</tr>
<tr>
<td>HD</td>
<td><code>&lt;embed&gt;</code></td>
<td>ED</td>
<td>no*</td>
<td>no*</td>
</tr>
<tr>
<td>ED</td>
<td><code>&lt;iframe&gt;</code></td>
<td>HD</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>ED</td>
<td><code>&lt;object&gt;</code></td>
<td>HD</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>ED</td>
<td><code>&lt;embed&gt;</code></td>
<td>HD</td>
<td>yes(DOM)</td>
<td>yes(DOM)</td>
</tr>
<tr>
<td>ED</td>
<td><code>&lt;iframe&gt;</code></td>
<td>HD</td>
<td>partial</td>
<td>partial</td>
</tr>
</tbody>
</table>

```javascript
function test_HD_A_iframe_ED_A_r0 {
    var id = getId();
    set(id, "no*"); // frame onload not executed
    var ee = document.createElement("iframe");
    ee.width=0;
    ee.height=0;
    ex.onload = function() {
        try {
            var svgDoc = ee.getSVGDocument();
            var firstChildName = "";
            svgDoc.documentElement.firstChild.nodeName;
            if (check if svg first ch/id name is "rect")
                set(id, firstChildNames==="rect")?yes(DOM):"no";
        } catch (ex) {
           set(id, "no*", ex.message); // SOP violation?
        }
    };
    ee.src = http://your-sop.com/img/svg.php?func=test_HD_A_iframe_ED_A_r0
    document.getElementById("loader").appendChild(ee); // load the content
   ...
```
# Your-SOP.com Testbed

You have detected 126 differences within 544 applicable test cases (23.16%).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>CANVAS with PNG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: your-sop.com Use-Credentials: true</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: (not set) Use-Credentials: (not set)</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: (not set) Use-Credentials: true</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: (not set) Use-Credentials: false</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: your-sop.com Use-Credentials: (not set)</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: your-sop.com Use-Credentials: true</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
<tr>
<td>HD</td>
<td>CANVAS with SVG</td>
<td>ED</td>
<td>Cross-origin: (not set) Access-Control-Allow-Origin: your-sop.com Use-Credentials: false</td>
<td>R</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>no</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
<td>yes(pixel)</td>
</tr>
</tbody>
</table>

17
Different Browser Behaviors

- >12%: Safari 9
  - Missing type: `image/svg+xml`
  - Fixed in Safari 10.1
- >35%: `<canvas>` and PNG/SVG (CORS)
- >51%: `<link>` (CORS)
- One IE/Edge vulnerability without using CORS
Cross-Origin Login Oracle Attack

<table>
<thead>
<tr>
<th>Description</th>
<th>Access-Control-Allow-Origin:</th>
<th>Access-Control-Allow-Credentials:</th>
</tr>
</thead>
<tbody>
<tr>
<td>embed ED</td>
<td>x yes yes yes yes yes yes no no yes yes</td>
<td></td>
</tr>
<tr>
<td>embed HD</td>
<td>r yes(DOM) yes(DOM) yes(DOM) yes(DOM) yes(DOM) no no yes(DOM) yes(DOM)</td>
<td></td>
</tr>
<tr>
<td>embed HD</td>
<td>w yes(DOM) yes(DOM) yes(DOM) yes(DOM) yes(DOM) no no yes(DOM) yes(DOM)</td>
<td></td>
</tr>
<tr>
<td>object HD</td>
<td>r partial partial partial partial partial partial partial no no partial partial</td>
<td></td>
</tr>
<tr>
<td>object HD</td>
<td>w partial partial partial partial partial partial partial no no partial partial</td>
<td></td>
</tr>
<tr>
<td>embed HD</td>
<td>r partial partial partial partial partial partial partial no no partial partial</td>
<td></td>
</tr>
<tr>
<td>embed HD</td>
<td>w partial partial partial partial partial partial partial no no partial partial</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>r no no no no no yes yes no no no no</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>w yes yes yes no no yes yes yes yes yes yes</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>r no no no no no yes yes no no no no</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>w yes yes yes no no yes yes yes yes yes yes</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>r no no no no no yes yes no no no no</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>w yes yes yes no no yes yes yes yes yes yes</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>r no no no no no yes yes no no no no</td>
<td></td>
</tr>
<tr>
<td>LINK ED</td>
<td>w yes yes yes no no yes yes yes yes yes yes</td>
<td></td>
</tr>
</tbody>
</table>
Cross-Origin Login Oracle Attack

• Webserver delivers different CSS files
  – User *logged in* or *logged out*?
• *a.com* attacks *victim.com*
  – `<link type="text/css" rel="stylesheet" href="//victim.com/style.css"/>
  – `<script>alert(document.styleSheets[0].cssRules[0].cssText)</script>`
Cross-Origin Login Oracle Attack
3. Limitations & Access Control Policies
Limitations

• 15 HTML elements with `src` attributes
  – Several more with a similar functionality
• Many sandbox attributes, ways to embed a document, MIME types, and pseudoprotocols
• `<link>`: imports, worker
• `<svg>`: JavaScript via xlink
• Growing surface with each new feature
Access Control Policies

- Discretionary Access Control (DAC)
- Role-Based Access Control (RBAC)
  - Enhanced RBAC
- Attribute-Based Access Control (ABAC)
4. Conclusions & Future Work
Conclusions & Future Work

• Different browser data sets to identify inconsistencies (edge cases are important)
• Discussion about access control policies may help to understand the SOP-DOM
• Future Work
  – Other SOP subsets, HTML elements/attributes
  – Pseudoprotocols
Thank you for your attention

✉ marcus.niemietz@rub.de
🐦 @mniemietz