XSS via Embedded Javascript in PDF and Possible DOM Access

After a detailed research I have came to the conclusion that even if a website is using PDF.js version 4 or below, it is not possible to access DOM properties via embedded JavaScript in a PDF file to steal cookies using document.cookie() via XSS (Cross-Site Scripting) attack. Here's a detailed explanation:

1. PDF.js Security Model

PDF.js is designed to safely render PDF files in the browser. The security model of PDF.js ensures that any JavaScript embedded within a PDF is executed in a highly restricted and isolated environment, preventing access to the parent document's DOM.

2. Sandboxing

PDF.js uses a sandbox to execute JavaScript embedded in PDF files. This sandboxing means that the JavaScript has no access to the DOM of the page that embeds the PDF viewer. Therefore, any attempt to execute document.cookie() within the embedded JavaScript would refer to the sandboxed document, not the parent web page.

3. Restricted JavaScript Execution

Even if JavaScript is embedded within a PDF, PDF.js restricts what this JavaScript can do. Typically, it's limited to simple PDF-specific tasks and cannot perform operations like accessing cookies, modifying the DOM, or making network requests.

4. No Direct DOM Interaction

JavaScript in PDFs, when executed by PDF.js, does not have direct access to the DOM of the embedding page. Therefore, it cannot call document.cookie() to steal cookies.

Glyph rendering and FontMatrix

Glyph rendering and **FontMatrix** are concepts related to how text and fonts are handled and displayed in PDFs, particularly within the context of pdf.js.

1. Glyph Rendering in pdf.js

Glyph rendering refers to the process of converting text characters into visual representations (glyphs) on the screen. In pdf.js, this involves:

- Parsing the PDF content stream to extract text information
- Mapping characters to glyphs using the embedded fonts or standard fonts
- Rendering the glyphs on a canvas element, which is how pdf.js displays the text content of PDFs

2. FontMatrix in pdf.js

A FontMatrix is a transformation matrix used to scale, translate, rotate, or skew the glyphs from a font to properly position and size them on the page. In pdf.js, the FontMatrix is used to:

- Adjust the size and position of glyphs based on the font's metrics
- Transform the coordinate space to ensure the text appears correctly on the canvas

Accessing DOM Properties with Embedded JavaScript in PDFs

When it comes to embedding JavaScript in PDFs, here are some key points to consider:

1. PDF JavaScript Environment

The JavaScript environment in PDFs are highly restricted and does not provide access to the browser's DOM. Its primary functions are related to form handling, document manipulation, and simple interactivity within the PDF itself.

2. pdf.js and JavaScript

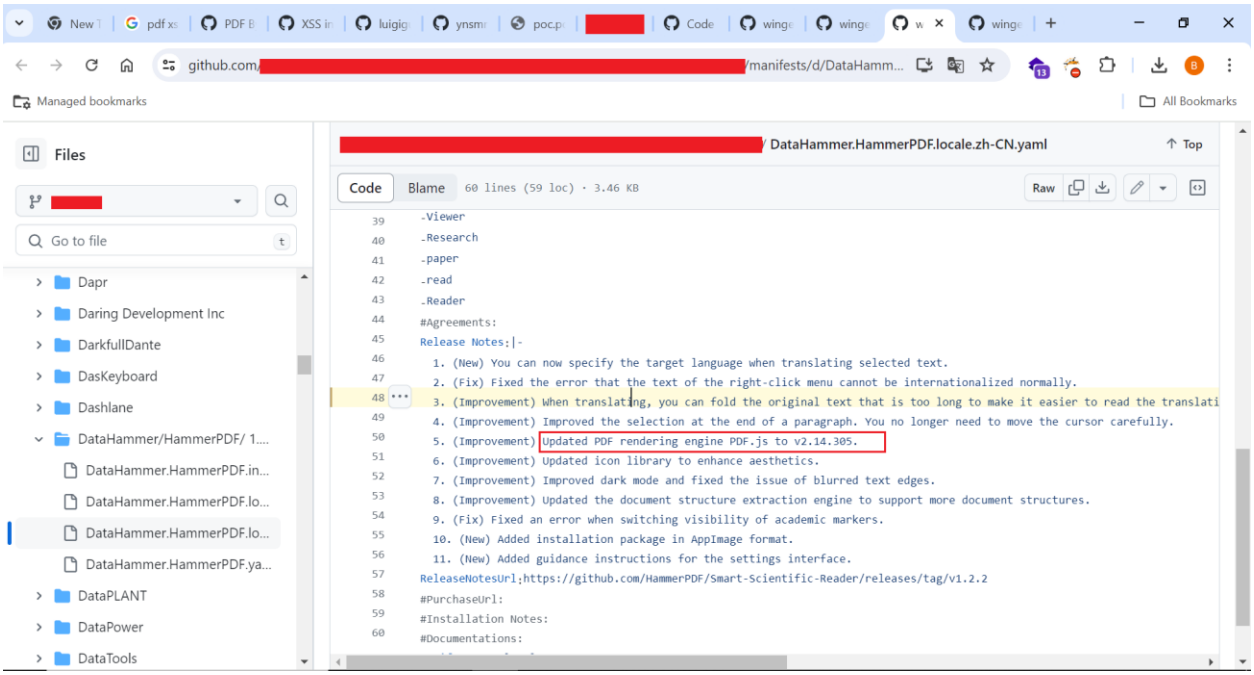
pdf.js is a JavaScript library used to render PDFs within a web application. While pdf.js can parse and render PDF content, it also does not support running embedded JavaScript from the PDF in a way that would interact with the DOM. Instead, pdf.js runs entirely within the context of the web application and its DOM.

Conclusion (based on research and analysis)

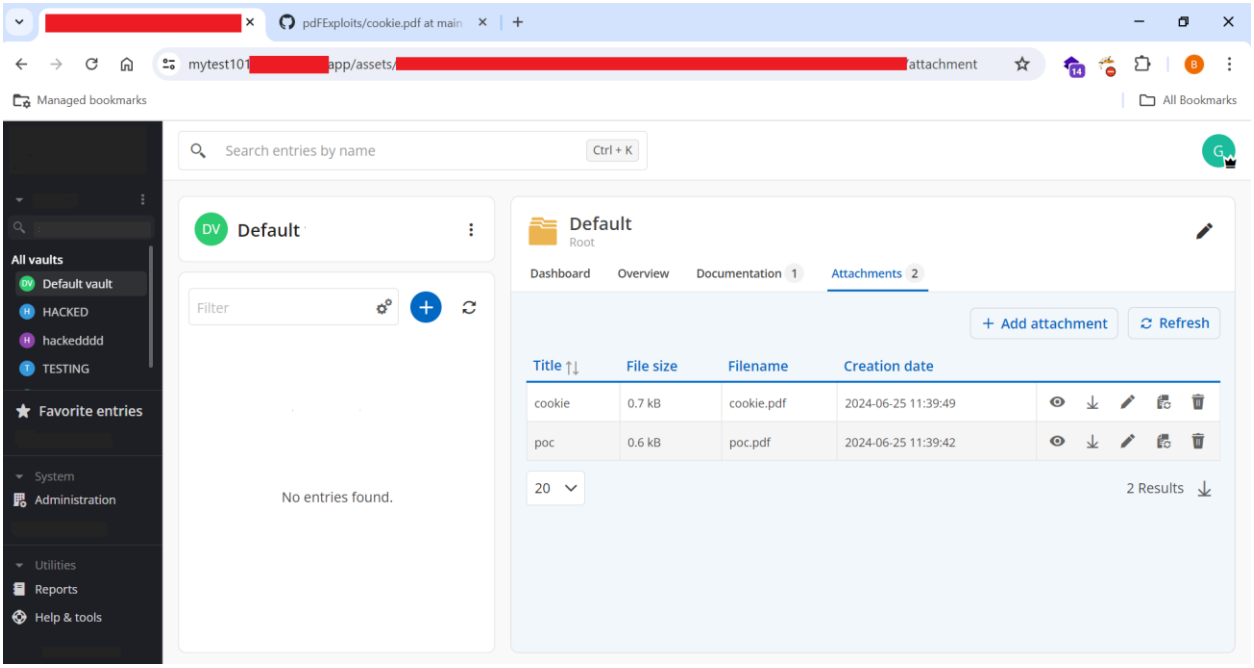
It is not possible to access DOM properties by embedding JavaScript into a PDF using glyph rendering, FontMatrix, or any other means. The JavaScript execution environment in PDFs is sandboxed and limited to functions related to the PDF itself, and pdf.js does not support running embedded JavaScript in a way that would interact with the browser's DOM. The primary role of pdf.js is to render the PDF content on a canvas within a web page, not to execute the PDF's embedded scripts in a broader web context.

Proof of Concept

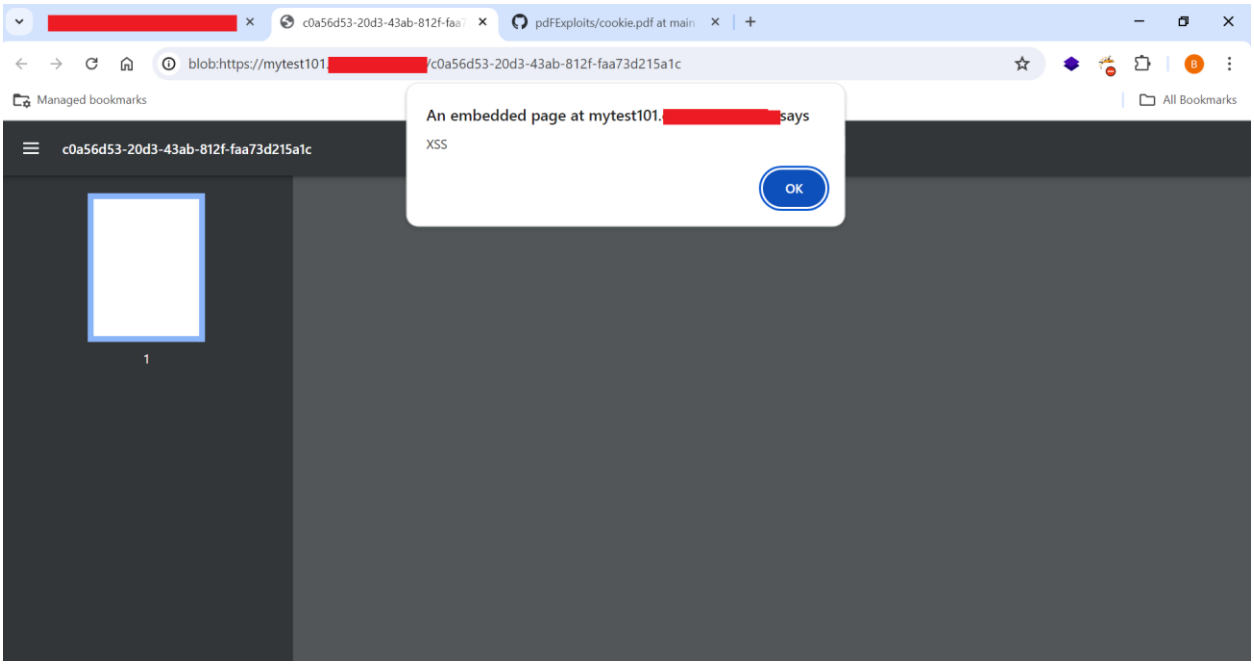
After the research I tried to reproduce the same vulnerability on one of the program I had which was using **pdf.js** with the version **2.14.305,** but even after trying different techniques along with the exploits provided on https://github.com/coffinxp/pdFExploits it was not possible to access the DOM properties and execute the ***document.cookie()*** function to steal the cookies by embedded javascript in PDF.



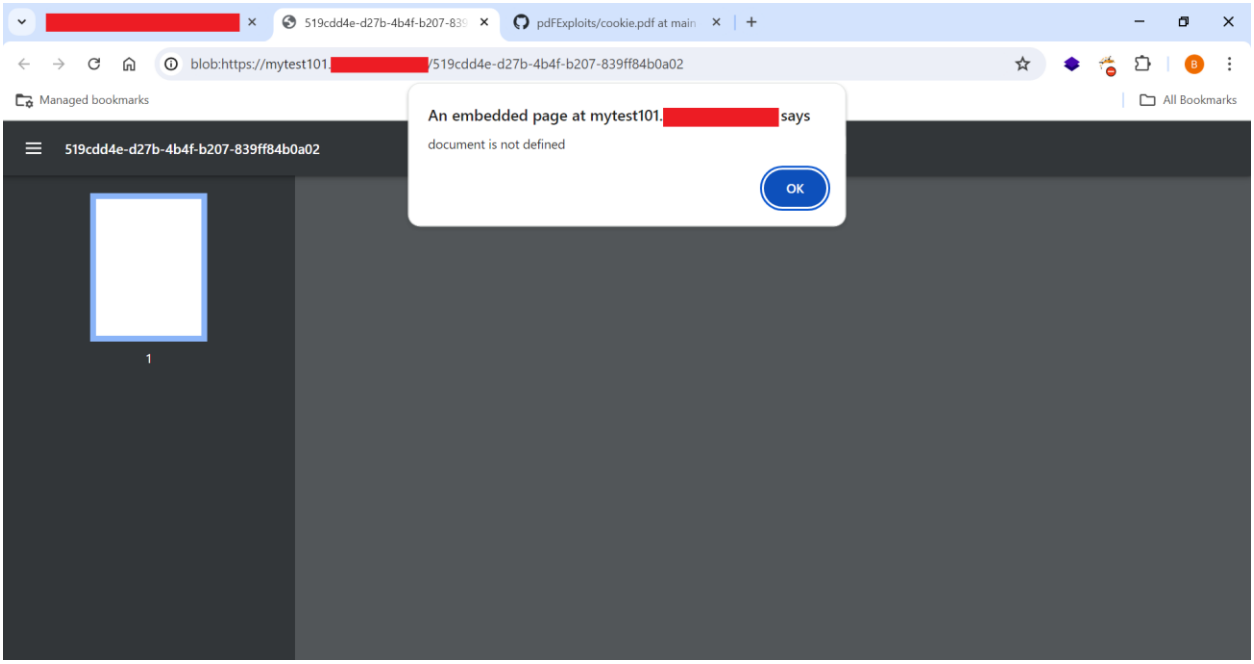
Step 1: I uploaded 2 files on my target application, poc.pdf and cookie.pdf where poc.pdf was having a normal **app.alert\("XSS"\)** and cookie.pdf had **app.alert\(document.cookie\)** functions.



app.alert\(“XSS”\)



app.alert\(document.cookie\)



NOTE: As you can see in the above screenshot the message says “document is not defined” which means it is not possible to access DOM properties by embedding Javascript in PDF.

In the below document you can see the pop-up with a cookie now let me explain how I did that, I simply copied the cookies from the web application and pasted it in **app.alert\("<Cookie_Values>"\)** so it showed as a text like it did in **app.alert\("XSS"\).** So now it is pretty obvious and sensibe how the cookie values or domain name got reflected in https://youtu.be/VxgEYQyx5EU?si=iNpVtp6TXzkHejp0 video given by https://x.com/coffinxp7/status/1804698234935509388?t=M71FTQkgd9V22fpp3tiyqQ&s=03