XSS in tagify's template wrapper #988

Closed 3 tasks done **rrott** opened this issue on Feb 16 · 7 comments

Labels **Clarification required**

rrott commented on Feb 16

Prerequisites

- [x] I am running the latest version
- [x] I checked the documentation and found no answer
- [x] I checked to make sure that this issue has not already been filed

Demo Page

React issue template:
tagify-react-wrapper-forked

Explanation

We have found a bug in tagify's template wrapper that leads to XSS vulnerability, making applications that use `tagify.js` or `react.tagify` vulnerable as well.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Refs:

https://owasp.org/www-community/attacks/xss/
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Affected file:

https://github.com/yairEO/tagify/blob/master/src/parts/templates.js#L13

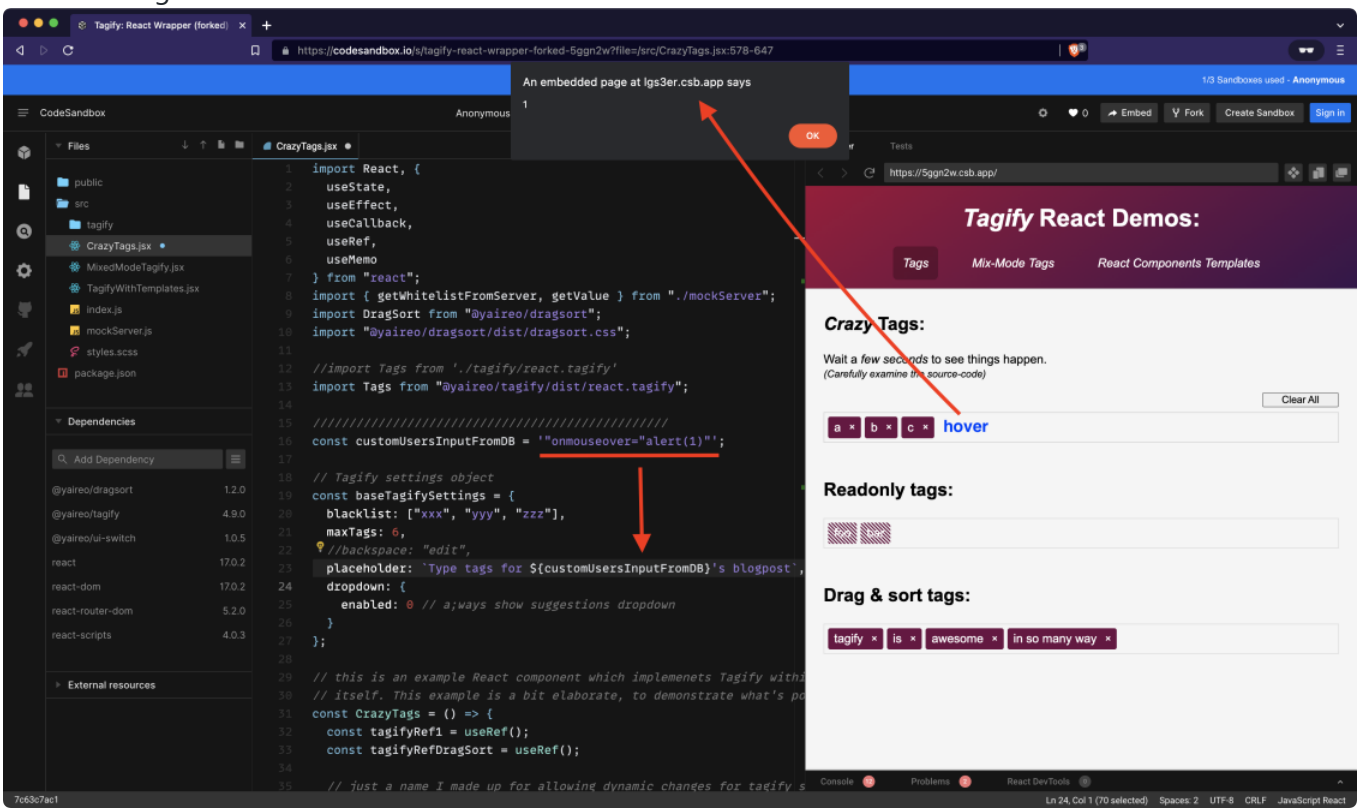
```
data-placeholder="${_s.placeholder || '&#8203;'}"
aria-placeholder="${_s.placeholder || ''}"
```

Example on codesandbox.io

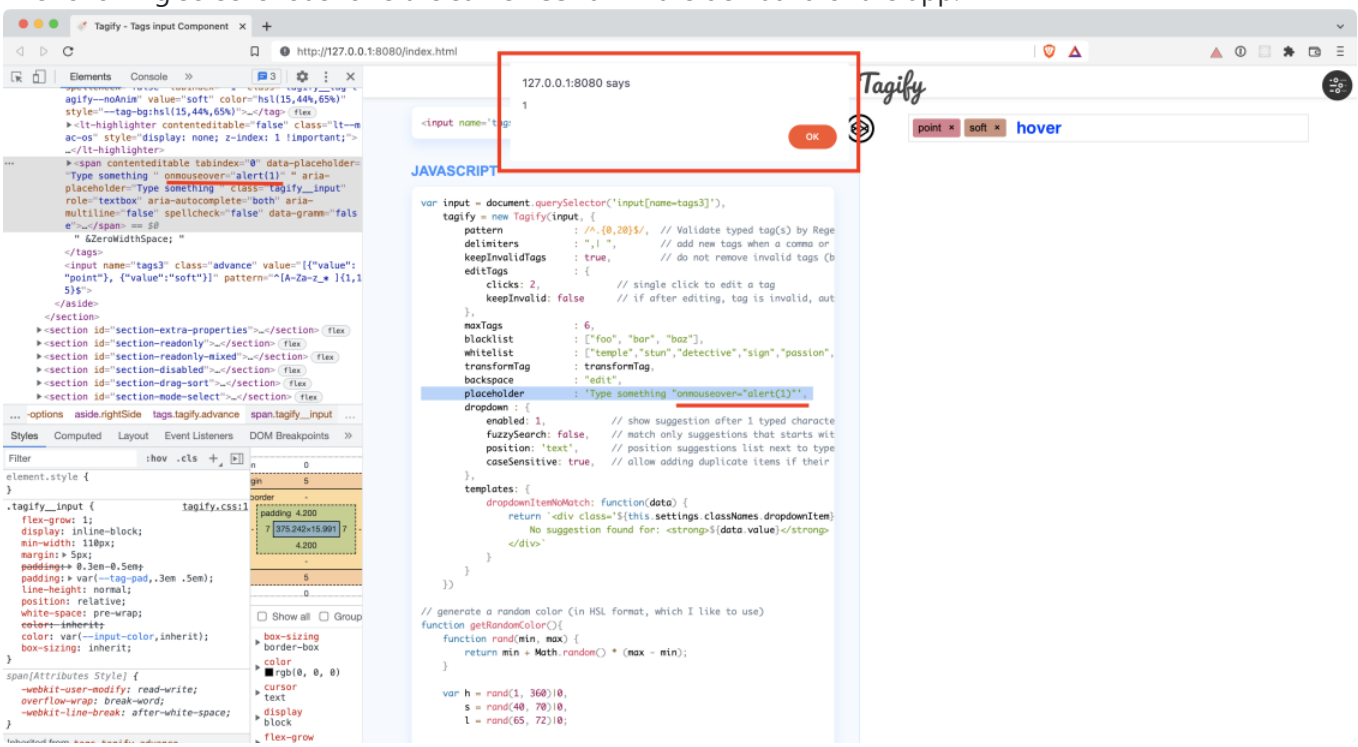
Steps to reproduce:

1. Open the following forked Tagify's React Wrapper demo
2. Notice line #17 where a `customUserInput` variable is declared. This variable mocks data that came from an API or an input.
3. On the line #23 we use the `customUserInput` variable to customize tags.
4. Once the demo app is rendered, open the "Tags" tab and hover on the first input. It will fire the XSS.

The following screenshot shows the XSS run on codesandbox.io



The following screenshot shows the same XSS run in the dev build of the app.



As you see, the tagify builds a new span with an attribute that was not supposed to be there.

  **rrott** mentioned this issue on Feb 16

fix for XSS in tagify's template wrapper #989

Closed

 **yairEO** commented on Feb 16 • edited Owner

Hi rrott, thank you for the detailed analysis

Why is this a Tagify issue?

It's up the the developer which uses Tagify to sanitize the content before saving it in the database.

front-end code assumes all content sent from the app's API (and from DB to API) has already been verified before it was saved to the DB.

1

  **yairEO** added the **Clarification required** label on Feb 16

rrott commented on Feb 16 • edited

Author

0. all developers make mistakes. It is better to supply them with more reliable and secure tools and libs.\
1. tagify's API does not provide any documented options to add onhover/onclick/etc handlers using the `placeholder` prop. There is no way to add the handlers using any other props described in the `TagifyWrapper.propTypes` object, except `placeholder`. It is undocumented, unintended, and unexpected behavior, so it is definitely a bug. And this bug leads to the XSS vulnerability.
2. Tagify builds the malicious `<span/>` tag and adds it to the DOM cause it uses untrusted input without any validation and sanitization. Never insert untrusted data and attribute encode before inserting untrusted data into HTML common attributes
3. database and API are not the only sources of malicious data.
4. developers do not know about this "feature" and may send unsanitized data to the component because they trust you. Additionally, this lib can be used as a part of another library, theme, or framework, and the end-users who do not know what lib they use for tags are affected as well
5. it is always better to have sanitization on both front-end and back-end parts altogether.

*as for the point 1) - it's not only about adding a custom attribute. It is about adding any HTML or JavaScript code and running it in the users' browsers.

1

yairEO commented on Feb 16

Owner

It is highly unlikely a developer will write javascript, intentionally, within the `placeholder` setting, which will harm the users, and if such developer does this, then that developer can basically do a million other things, because they are the author of their program.

If a developer writes this, then he/she did this on purpose with full knowledge what will happen.

Also, there are custom templates which a developer might use, so by your logic, the developer can also add any javascript there as well, so adding stuff to the HTML can happen in several places where the developer has control over, for example:

```
let tagify = new Tagify(inputElm, {
    placeholder: 'x "onmouseover=\"alert(1)\""',
    templates: {
        tag(tagData, tagify){
            var _s = this.settings;
            return `<tag title="${(tagData.title || tagData.value)}"
                        contenteditable='false'
                        spellcheck='false'
                        tabIndex="${_s.a11y.focusableTags ? 0 : -1}"
                        class="${_s.classNames.tag} ${tagData.class || ""}"
                        ${this.getAttributes(tagData)}>
                <x title='' class="${_s.classNames.tagX}" role='button' aria-label='remove tag'></x>
                <div onmouseover=\"alert(1)\">
                    123
                </div>
            </tag>`
        },
    },
```

Can you please tell me how would such malicious code might gets to those settings without a developer intentionally wrote it?

rrott commented on Feb 16 — Author

I've created the PR 989, which uses your internal helper, designed precisely for that purpose.
The fix:

- does not add anything new.
- does not make code unreadable or unsupportable.
- breaks nothing.
- And it fixes the XSS.

I cannot understand why you tend to reject it?

Here is an example of such a typical behavior: tagify-react-wrapper-forked
E.g., developers may want to personalize placeholders, adding custom text using template literal. They do not wish to hard-code something malicious; all they need is to make their app fancy. The variable can be either taken from a database, API, 3rd party library, or anywhere else - it does not matter. The problem is that Tagyfy gets untrusted input, builds the malicious code, and puts it directly into the DOM.

```
// it can be a result of a previous fetch to a 3rd party API
const customUserInput = '"onmouseover="alert(1)"';

// Tagify settings object
const baseTagifySettings = {
  blacklist: ["xxx", "yyy", "zzz"],
  maxTags: 6,
  //backspace: "edit",
  placeholder: `add tags for the ${customUserInput}'s blogpost`,
  dropdown: {
    enabled: 0 // a;ways show suggestions dropdown
  }
};
```

An attacker, not the developer, could put the XSS payload.

All the breaches, hacks, and data losses result from somebody's mistake. Many developers use StackOverflow, copy-paste code, skip testing, etc., etc. None is perfect - even Tagify has a bug and unexpected behavior.

> so by your logic, the developer can also add any javascript there as well, so adding stuff to the HTML can happen in several places where the developer has control over

No, the logic is: developers do not expect Tagify to parse and render XSS payloads. It is undocumented and dangerous behavior.

yairEO commented on Feb 17 — Owner

I've seen your PR and the fix was done in the wrong place, since many developers are using custom templates, which your fix does not address.

> I cannot understand why you tend to reject it?

Because I do not agree with your level of panic and I believe none of what you've said will ever happen. Nobody ever will use a placeholder from a 3rd-party script, while that is possible, it will not happen.

I just don't understand what do you get from all this and why are you doing all this? what is you agenda? are you randomly going over open-source code and checking for potential vulnerabilities, or are you a Tagify user?

rrott commented on Feb 17 — Author

Check my acc to see that you are wrong. I saw this bug in the wild, in a real project. The issue's root was in Tagify, so PR was sent to improve it. It's open-source, is not it? I came up with a bug report and a solution; nobody blames the lib.

Let's discuss the PR. I am unfamiliar with Tagify's codebase, and I can be wrong.
What can I do to improve it? Do you want me to move sanitization to any other place?
I thought that as the root of the issue is injecting payloads in the wrapper function, the fix should be there. As I understand, this `wrapper` is a built-in standalone template, but developers can write any other custom template. They definitely may write buggy and vulnerable templates, and it is hard to prevent them from doing this, but their code is under their control, while `wrapper` comes with the lib and can't be updated/fixed without monkey-patching or forking the whole lib. That's the point.

👍 1

yairEO commented on Feb 17 • edited — Owner

I've implemented your suggestion, but in a better way, and will push the fix soon

👍 4 🚀 1

yairEO closed this as completed on Apr 5

Assignees

No one assigned

Labels

Clarification required

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants