RCE via stored XSS in note export rendering

Critical **thecodrr** published **GHSA-fjm8-jg78-89h4** last week

Package

Notesnook Web/Desktop (Notesnook)

Affected versions	Patched versions
< 3.3.15	>= 3.3.15

Notesnook iOS/Android (Notesnook)

< 3.3.20	>= 3.3.20

Description

Summary

A stored XSS in the note export flow can be escalated to remote code execution in the desktop app.

The root cause is that exported note fields such as `title`, `headline`, and `content` are inserted into the generated HTML template without HTML escaping. When the note is later exported to PDF, Notesnook renders that HTML into a same-origin, unsandboxed iframe using `iframe.srcdoc = ...`. Injected script executes in the Notesnook origin. In the desktop app, this becomes RCE because Electron is configured with `nodeIntegration: true` and `contextIsolation: false`.

Details

The export flow embeds note-controlled data into an HTML template without escaping:

- `packages/core/src/collections/notes.ts:416`
  - `notes.export()` passes note data into the HTML template builder.
- `packages/core/src/utils/templates/html/template.ts:31`
  - `headline` is interpolated into a meta tag.
- `packages/core/src/utils/templates/html/template.ts:33`
  - `title` is interpolated into `<title>`.

- `packages/core/src/utils/templates/html/template.ts:210`
  - `title` is interpolated into `<h1>`.
- `packages/core/src/utils/templates/html/template.ts:211`
  - `content` is interpolated directly into the HTML body.

The exported HTML is then rendered for PDF generation through an unsandboxed `iframe.srcdoc`:

- `apps/web/src/common/export.ts:40`
  - `exportToPDF()` creates an iframe and sets `iframe.srcdoc = content`.
  - No `sandbox` attribute is applied.

The data path from note content into exported HTML is reachable through the normal export flow:

- `packages/common/src/utils/export-notes.ts:219`
  - exported note content is prepared and passed into `database.notes.export()`.

On desktop, this becomes code execution because the main renderer is intentionally Node-enabled:

- `apps/desktop/src/main.ts:126`
  - `nodeIntegration: true`
  - `contextIsolation: false`
  - `nodeIntegrationInWorker: true`

Because `srcdoc` inherits the parent origin, JavaScript running inside the exported iframe can call:

```
parent.require("fs")
and use Node primitives such as fs, child_process, etc.

This means the vulnerability is not just stored XSS in exported HTML; in Notesnook Deskt

### Details
_Give all details on the vulnerability. Pointing to the incriminated source code is very

### PoC
1. The attacker prepares a malicious note title containing HTML/JavaScript. For example:

   ```html
   </h1><script>const fs=parent.require("fs");const os=parent.require("os");const path=p
The attacker gets this note into the victim’s Notesnook account. This can happen through

The victim opens the malicious note in Notesnook.

The victim exports the note as PDF.

During export, Notesnook converts the note into HTML and inserts attacker-controlled fie

Notesnook then renders the generated export HTML into a same-origin iframe using iframe.

Because the iframe is not sandboxed, the injected script executes in the Notesnook rende

In the desktop app, Electron is configured with nodeIntegration: true and contextIsolati
```
```

```
This allows the attacker to execute arbitrary local code in the victim’s desktop app con
Expected result:
<img width="1919" height="1079" alt="image" src="https://github.com/user-attachments/ass
xss via exported note title
This confirms that JavaScript embedded in the note title executed during export and acce

A minimal alert-based payload for confirming JavaScript execution is:

</h1><script>parent.alert(1)</script><h1>
<img width="1919" height="1026" alt="image" src="https://github.com/user-attachments/ass

However, the file-write PoC is more reliable evidence because it proves code execution r
case rce :
Create a new note.
Set the note title to:
<script>
const { exec } = parent.require('child_process');
exec('calc.exe', (error) => {
  if (error) console.log('Error:', error);
});
document.body.insertAdjacentHTML('beforeend', '<p id="executed" style="color:red;font-si
</script>
Set the body to any benign text, for example: test.
Export the note as PDF.
Check whether the following calc open
<img width="1919" height="1035" alt="image" src="https://github.com/user-attachments/ass

### Impact
This is a stored client-side injection that becomes arbitrary code execution in the desk

Who is impacted:

Notesnook Desktop users who export an attacker-controlled note.
Any workflow that imports or opens untrusted note data before export is in scope.
Security impact:

Arbitrary local code execution in the victim’s Notesnook Desktop context
Access to Node/Electron APIs from attacker-controlled JavaScript
Potential file read/write and command execution on the victim machine
Full compromise of the desktop app session and any data accessible to that process
```

Severity

Critical 9.6 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None

User interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H

CVE ID

CVE-2026-42090

Weaknesses

- CWE-79
- CWE-94

Credits

iiihaiii	Reporter
01zulfi	Remediation developer
thecodrr	Remediation verifier