Exploit Title: Chatbox is affected by cross-site scripting (XSS). An attacker has to upload any XSS payload with SVG, XML file in Chatbox.

#There is no restriction on file upload in Chatbox which leads to stored XSS

Date: 19/12/2020

Exploit Author: Ritesh Gohil

Vendor Homepage: https://getgist.com/

Software Link : https://getgist.com/live-chat-software/

Version: 1.0

Tested on: Kali Linux and Windows 10

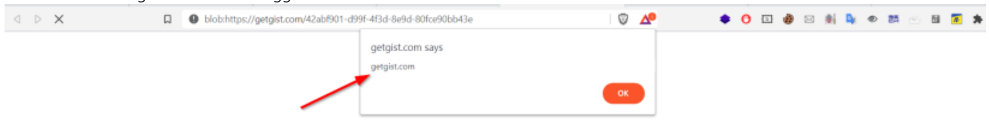
Contact: https://www.linkedin.com/in/riteshgohil25/

Stored XSS

Attack Vector: This vulnerability can results attacker to inject the XSS payload into the IMAGE URL and each time any user will go to that URL, the XSS triggers, and the attacker can able to steal the cookie according to the crafted payload.

Steps to Reproduce:

1. Take Chatbox Demo from https://getgist.com/
2. In the Chatbox. open it and start conversation.
3. After observing chatbox, I have found that there is no restriction on upload files. I can able to upload xss payload with SVG, XML ... many more.
4. For example, I have uploaded xss payload with SVG file. (Note Inside the payload you can use: alert(document.location) , alert(1), alert(document.domain) >> I have used in POC)
5. Boom Click on Image XSS has been triggered.



blob:https://getgist.com/42abf901-d99f-4f3d-8e9d-80fce90bb43e

getgist.com says

getgist.com

Payload: XSS Payload with SVG or XML File.

Thank you

IMPACT:

The impact of an exploited XSS vulnerability on a web application varies a lot. It ranges from user's Session Hijacking, and if used in conjunction with a social engineering attack it can also lead to disclosure of sensitive data, CSRF attacks and other security vulnerabilities.

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can:

Perform any action within the application that the user can perform. View any information that the user is able to view. Modify any information that the user is able to modify. Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

Mitigation:

Filter input on arrival : At the point where user input is received, filter as strictly as possible based on what is expected or valid input. Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

Use appropriate response headers. : To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

Content Security Policy : As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.