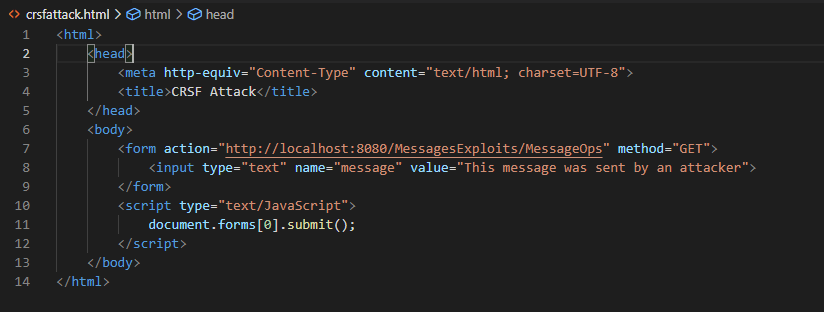
**Web Application Security (ITSC 302)**

**Assignment 1: CRSF, XSS, and DOM Attacks**

**Submitted by Coleton Sanheim**

**CRSF Attack:**

Here I utilized a simple html crsf attack to force a logged in user to send a message.



All a user needs to do is to visit this malicious page while logged in to the application and a message saying “This message was sent by an attacker” is sent to the message board under the user’s name. This same method can be utilized for different uses by simply changing the URL and the input to match what you want to affect.

Here is the result of a user visiting the page above.

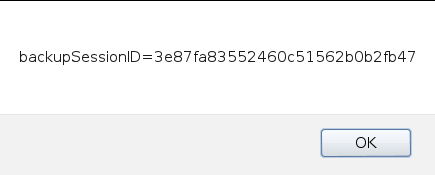


**XSS Attack:**

In this attack I utilized an XSS attack to steal the sessions cookies.

<script type="text/JavaScript">alert(document.cookie);</script>

This is a very simple script injection that will grab the current session’s cookies and display the cookies grabbed in an alert. All this requires is a text field that you can enter the above line into. In this case since we are using a message board, this message is persistent and any user that access this page will have the cookies displayed as an alert. This attack can be altered to grab any specific information that an attacker may want access to.



The alert that is displayed.

**DOM (HTML injection) Attack:**

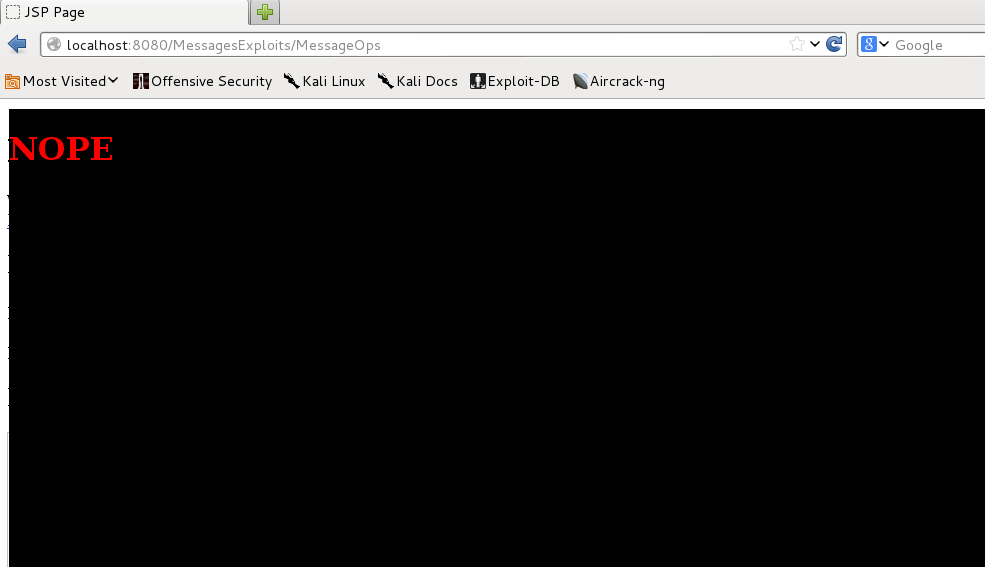
Finally for this attack I created a DOM html injection that very simply covers the whole page with a black square.

<div style="position:absolute;top:10px;left:10px;height:1000px;width:1000px;background-color: black;">

<h1 style="color:red">NOPE</h1>

</div>

Above is the code that was injected. It creates a div and sets certain parameters to set the location to start at the top right of the page, the size to be big enough to cover the whole page and the color to be black. It then prints a message saying “NOPE” over the blacked-out page. This acts as a denial of service as any users attempting to interact with the page will be unable to do so as the cannot access any elements of the application. This can be easily changed to any form of html elements.



Here is what the users are subject to upon logging into the page.

**Recommendations:**

To mitigate further CSRF attacks I would recommend implementing CSRF tokens, that (as far as I understand them) creates a specific token hidden in the server that is unique to that user’s session and will only accept input from that user’s session and will not accept any requests from a different session, even if the user is currently authenticated. (Note that this defense can be defeated by XSS attacks so make sure that you also implement XSS mitigation.)

To mitigate XSS attacks one simple method is to use input filtering, to prevent users from inputting certain characters or strings that are commonly used in JS scripts. Or if you want to allow your users more freedom, you could encode those characters when they are read into your application so that any script that would have been run is now invalid (IE. A < symbol is converted to &lt).

To mitigate DOM and html injection attacks the best method is to utilize similar methods to that of XSS mitigating as described above.