CS558 Lab 4

Konstantino Sparakis
BUID: U471315872

March 31, 2017

Late Days Used For Lab4: 0
Total Late Days: 2
Collaborators: None
Sources:

https://www.kirupa.com/html5/making_http_requests_js.html
http://michael-coates.blogspot.com/2008/06/cross-site-scripting-blacklist-vs.html
https://stackoverflow.com/questions/10730362/get-cookie-by-name
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://excess-xss.com/
https://stackoverflow.com/questions/17940811/example-of-silently-submitting-a-post-form-csrf

2.0 (csrf_0.html) : In order to execute this attack I use a Iframe and a form submission. The Iframe hosts the other page within this page so it gets by the Cross Origin Request Policy that can be annoying. Using some Jquery I also hide this, so there is no evidence of the attack. The form we submit contains all the information about login that would be needed such as attack password, username, and we pass the security level settings as well.

2.1 (csrf_1.html) : This is like a nested attack, we use xss exploit to run code that retrieves the token needed then post it making it a csrf, because we force the user to login as the attacker. When the webpage opens it executes a search query that runs my malicious code. I accomplish this by using a function called get_cookie() that I found in stackoverflow, which is referenced above, to get the token we need. Once I have that token, I link jquery to make an ajax post request that logs in the user as to the attacker account passing in the token making it valid. I also use an iframe to execute this and using some Jquery I also hide this, so there is no evidence of the attack.

3.0 (xss_0.html) : This is a very simple attack, once the page opens it opens within the iframe so the user doesnt know it even happened. The link I use for my get request, exploits the fact that its url input is not sanitized for scripts. so http://bungle.com/search/q? <add script content here>. I found issues just using a <script> tag so instead I used a <body onload=""> function to simply call an alert(document.cookie).

3.1 (xss_1.html) : Building on 3.0, I do everything the same except I use a <script> in this one. I run the netcat server with the following command "nc -l -p 31337", I ran into issues when I did not include the -p. inside the script tags, I link jquery so that I can execute a get request containing the document.cookie and send that to the server. I had issues figuring out how to close the connection so the server hangs once it receives this.

3.2 and 3.3 (xss_2,3.html) : I use the same attack to solve both 3.2 and 3.3. After some pondering and googling I found an interesting blog pointing to that because we remove "script", if we layer it such as <scriscriptpt>, the function removes the inside "script" leaving us with just script which executes the attack. I used XMLHttpRe-

quest() function here in order to avoid loading jQuery to do my post. I run the netcat server with the following command "nc -l -p 31337" again as without the -p it doesn't work on my local machine, I also had issues with setting the header to connection close, which you can see I do in the code but apparently is not supported and doesn't actually execute.