Cookies and Cross-site Scripting

Part 1: Cookies

a. There is a single cookie whose name is “theme” and whose value is “default”

b. There is still a single cookie with the same name, but the value is now “red”

c. Under the “Request cookies” section, the name and value fields are the same as in the previous steps, but if I look under the “Request headers” section, there is a header named “Cookie” that corresponds to a value of “theme=default” which is also how it appears in the raw intercept body.

d. The red theme is where I left it and upon relaunching, it was still set to the red theme and the request cookies still list the name of the cookie as “theme” with the value still listed as “red.”

e. The current theme is transmitted through an HTTP GET request, which allows us more visbility than a POST request.

f. When I change the theme, this is also transmitted through an HTTP GET request. The part of the cookie that changes is the value. So far, I’ve only been switching between red and blue, so the value of the cookie being sent is always the color I’m changing the theme to.

g. I changed the HTTP Get request in the “Network” section under the “Headers” tab that popped up after I clicked on the GET request corresponding to the theme.

h. I edited the GET request and the cookie (both were necessary), then pressed the “Forward” button to change the theme successfully.

i. Mac OS (the OS I am using) stores Cookies in a local file specific to the browser that the Cookies belong to. For Google Chrome (my browser of choice), the path is “~/Library/Application Support/Google/Chrome/Default/Cookies”

Part 2: Cross-Site Scripting (XSS)

a. It appears that Moriarty simply added a javascript alert trigger to the page that executes upon opening his post

b. Given that Moriarty was able to get my browser to execute code, there's a lot of worse damage that he could've done. He could've used Javascript to read sensitive information in my Cookies, for example, since some websites use Cookies for authentication (which is why we have to clear them and the cache every so often when we're having issues with a website or the browser at large).

c. Moriarty could gain access to my system information, such as details about my browser type, version, and my OS, which could make my system vulnerable to further, more specific attacks that have a better chance at being successful.

d. If I wanted to prevent Moriarty from triggering an alert, I could change my browser's settings to disallow some javascript features or I could install a browser extension with an event listener that can listen for and intercept an alert.