Miles Hoene-Langdon

Part 1: Cookies

a. There is 1 cookie for http://cs338.jeffondich.com/fdf/. The name of this cookie is theme and value is default.
b. Yes, the value of the cookie named “theme” changes to “blue” when blue is selected in the theme tab and “red” when red is selected.
c. Using burpsuite I see that HTTP requests have packets with the header “Cookie:” with a value of “theme=default”, “theme=red”, or “theme=blue” depending on the current Cookie value. If the requested change differs from the current theme, the HTTP response has a “Set-Cookie:” header with values of “theme=default”, “theme=red”, or “theme=blue” depending what value is requested in the HTTP request. This is then followed by the expiration of the cookie. I do see the same values for the cookies that I did with the inspector.
d. Yes, the red theme that I had before I quit the page is still selected.
e. The current theme of “red” is transmitted between the browser and the ftf server through the cookie saved by my browser. When I come back to the page, my browser sends the cookie named “theme” with its value of “red” to the server with my HTTP request for the homepage of ftf. The HTTP request is provided below.

```
GET /fdf/ HTTP/1.1
Host: cs338.jeffondich.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.105 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: theme=red
Connection: close
```

f. The change to theme is transmitted between the browser and the FDF server through the query in the HTTP request. The packet to change the theme from red to blue is shown below. In this pack we see “GET /fdf/?theme=blue HTTP/1.1”. This query string beginning with a question mark is how the server sees that the client wants to change the theme.

```
GET /fdf/?theme=blue HTTP/1.1
Host: cs338.jeffondich.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.105 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://cs338.jeffondich.com/fdf/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: theme=red
Connection: close
```

g. In the inspector you can access the cookies stored in your browser and change their values. Thus, the theme can be changed by going into the inspector and changing the value of the “theme” cookie to one of the possible values and then reloading http://cs338.jeffondich.com/fdf/.
h. In Burp Suite’s proxy tool you can intercept a “GET /fdf/ HTTP/1.1” request and change the value of “theme” under the “Cookie:” header to one of the other possible values. Then, forwarding the edited intercepted packet will successfully change the theme without the use of FDF’s Theme Menu.

i. Kali stores its cookies for firefox at file path “~/mozilla/firefox/<profile path>/cookies.sqlite”. These cookies can be read using sqlite3.

Part 2: Cross-site Scripting

a. Moriarity begins by making the two posts. The first post deliberately includes an HTML span tag “red text”. The website takes in the body of the post and places it directly into the html file for the post webpage. Now when a user clicks on Moriarity’s first post, the browser receives the html file for the page and processes the span tag. This causes “red text” to be presented in red to the user, in the body of the post. Similarly, the second post by Moriarity includes javascript with an alert that contains the message “Mwah-ha-ha-ha!”. The website then takes the body of this post and pastes it directly into the html file for the post webpage that is created. When a user of FDF now clicks on Moriarity’s second post, their browser receives the corresponding html file from the FDF server and runs it. Thus, the javascript is run and the user is immediately met with an alert containing the message “Mwah-ha-ha-ha!”.
b. An XSS attack that is more virulent than Moriarity’s attacks would be making a post containing an HTML script that opens a new window to a site containing malware. When a user clicks on this page, the server sends their browser the html file for the post. Then when the browser processes the script, the user is brought to a malicious website where malware or spyware could be installed on their system.
c. Another attack that is more virulent than Moriarity's attacks would be a post that contains javascript that makes the expiration of the user’s session cookie a time in the past. When a user clicks on this page, the server sends their browser the html file for the post. Then when the browser processes the script in the html file, the expiration of the user’s session cookie will be set to a time in the past. Thus, the cookie will no longer be valid and the user will be logged out of their account.
d. To prevent what Moriarity is doing, the server could prevent posts from containing the character “<” and “>”. Without these the post will not be able to contain html scripts, so Moriarity cannot change the color of their text or create alert windows. The server could also replace the “<” and “>” with “<” and “>” in posts like is done with the post source code display on FDF, so users can still use the angle brackets. The browser could also block alerts. This would prevent Moriarity’s second post from taking effect.