Cross-site scripting HW

Author: Prompt Eua-anant

Part 1

a) Go to FDF and use your browser's Inspector to take a look at your cookies for cs338.jeffondich.com. Are there cookies for that domain? What are their names and values
   i) Name: Theme
   ii) Value: Default

b) Using the "Theme" menu on the FDF page, change your theme to red or blue. Look at your cookies for cs338.jeffondich.com again. Did they change?
   i) The cookie value changed to ‘red’ when I changed the theme to red.

c) Do the previous two steps (examining cookies and changing the theme) using Burpsuite. What "Cookie:" and "Set-Cookie:" HTTP headers do you see? Do you see the same cookie values as you did with the Inspector?
   i) Changing the theme from red to blue:
      1) A GET request with cookie Theme:red is sent to the server
      2) Server responds with a 200 OK along with the header Set-Cookie: theme=blue
      3) Upon inspecting, the browser sends several GET requests with cookie Theme:blue to the server

d) Quit your browser, relaunch it, and go back to the FDF. Is your red or blue theme (wherever you last left it) still selected?
   i) Yes. Theme is still blue, and the cookie value is ‘blue’

e) How is the current theme transmitted between the browser and the FDF server?
   i) The browser’s GET request contains a header ‘Cookie’ with value ‘blue’. The server’s html includes the following line: <main class="container blue"> which presumably sets the theme to blue. Moreover, the server’s response also includes a ‘set-cookie’ header with value ‘blue’, with an expiration date of ‘Sat, 25 Jan 2025 00:44:42’. This stores the cookie information somewhere on the kali machine so next time it accesses the server, the cookie will be sent.

f) When you change the theme, how is the change transmitted between the browser and the FDF server?
   i) Browser sends GET ‘/fdf/?theme=blue HTTP/1.1 query’ to server
   ii) This means set the theme to blue
   iii) Server sends response with header ‘Set-cookie: blue’ and the html contains the line <main class="container blue">

g) How could you use your browser's Inspector to change the FDF theme without using the FDF's Theme menu?
   i) Click inspect and find the line <main class="container red">, then change red to blue or default.

h) How could you use Burpsuite's Proxy tool to change the FDF theme without using the FDF's Theme menu?

        i) Modify the request: GET '/fdf/?theme=[whatever theme you want - default,blue,red] HTTP/1.1 query' to server

i) Where does your OS (the OS where you're running your browser and Burpsuite, that is) store cookies? (This will require some internet searching, most likely.)
    i) Kali Linux Burpsuite: home/kali/.BurpSuite/pre-wired-browser/Default/Cookies

Part 2

a) Provide a diagram and/or a step-by-step description of the nature and timing of Moriarty's attack on users of the FDF. Note that some of the relevant actions may happen long before other actions.
    i) Moriarty sends a POST request to the server, which stores Moriarty's title and post on the server.
    ii) Some time has passed.
    iii) Someone else clicks on Moriarty's post, which sends a GET request to the server. Accordingly, the server sends an html with Moriaty's title and post, which will get processed by the browser. If the html contains Moriaty's script, the user's browser will execute that script.

b) Describe an XSS attack that is more virulent than Moriarty's "turn something red" and "pop up a message" attacks. Think about what kinds of things the Javascript might have access to via Alice's browser when Alice views the attacker's post.
    i) Obtain Alice's cookie information using the document.cookie variable, then include a GET or POST request to the attacker's server to send the information obtained.

c) Do it again: describe a second attack that is more virulent than Moriarty's, but that's substantially different from your first idea.
    i) A script that redirects to a fake login page that looks identical to the FDF's login page, where Alice may enter the username and password.

d) What techniques can the server or the browser use to prevent what Moriarty is doing?
    i) Server: replace special characters like < or > and replace it with the text version like &lt or &gt.
    ii) Browser: can check the protocols that Moriaty's script sends personal info, eg. if TLS is not used, then Moriaty's server has no certificate ergo may be unsafe, and Alice's browser should ping a notification that allows Alice to voluntarily execute or not execute the script. Alternatively, the browser can just preclude the possibility of executing the script at all.