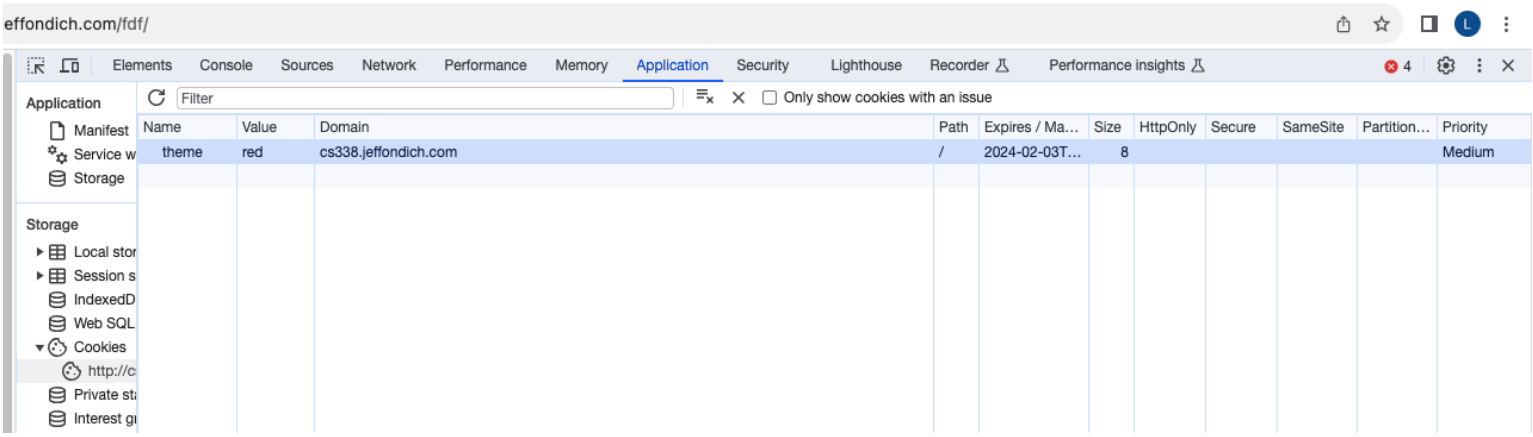
File name: hacking/xss.pdf
File owner: Lysander Miller
Collaborators: I brainstormed ideas with Kiri Salij

COOKIES AND CROSS-SITE SCRIPTING (XSS)

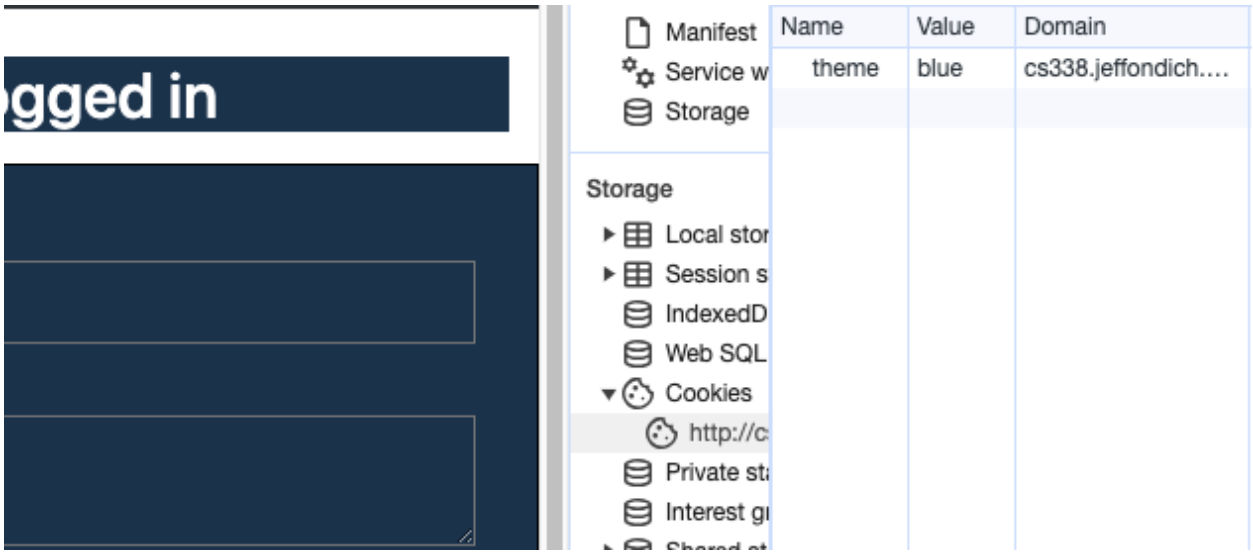
Part One - Cookies:

A.



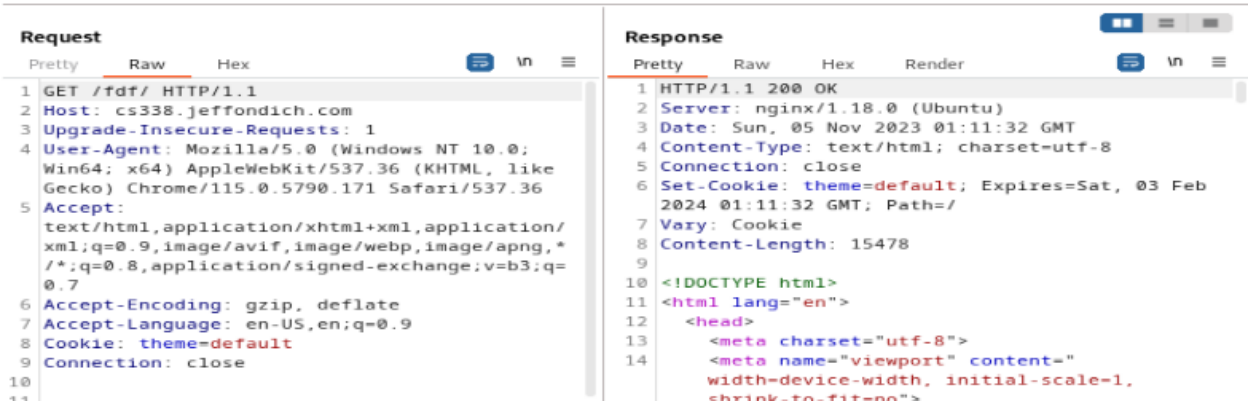
There is one cookie for cs338.jeffondich.com. Its name is “theme” and its value is “red”.

B.



The value of the cookie “theme” changed from “red” to “blue”.

C. When I went to http://cs338.jeffondich.com, I saw the following request and response with Burpsuite:



Request

Pretty Raw Hex \n

```
1 GET /fdf/ HTTP/1.1
2 Host: cs338.jeffondich.com
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0;
  Win64; x64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/115.0.5790.171 Safari/537.36
5 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
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: theme=default
9 Connection: close
10
```

Response

Pretty Raw Hex Render \n

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sun, 05 Nov 2023 01:11:32 GMT
4 Content-Type: text/html; charset=utf-8
5 Connection: close
6 Set-Cookie: theme=default; Expires=Sat, 03 Feb
  2024 01:11:32 GMT; Path=/
7 Vary: Cookie
8 Content-Length: 15478
9
10 <!DOCTYPE html>
11 <html lang="en">
12   <head>
13     <meta charset="utf-8">
14     <meta name="viewport" content="
     width=device-width, initial-scale=1,
```

As you can see above, the cookie “theme” starts out with the value “default”. Then, the http response sets the cookie “theme” to the value “default”.

After pressing the button to change the theme to red, I got the following request and response:

Request

Pretty Raw Hex \n

```
1 GET /fdf/?theme=red HTTP/1.1
2 Host: cs338.jeffondich.com
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/av
  if,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
  ;v=b3;q=0.7
6 Referer: http://cs338.jeffondich.com/fdf/?theme=default
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: theme=default
10 Connection: close
11
12
```

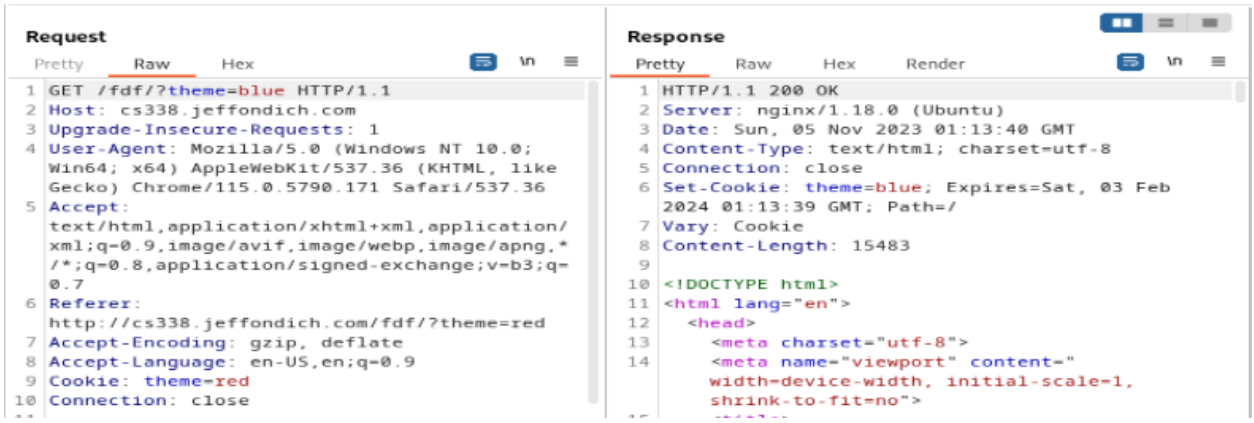
Response

Pretty Raw Hex Render \n

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sun, 05 Nov 2023 01:33:50 GMT
4 Content-Type: text/html; charset=utf-8
5 Connection: close
6 Set-Cookie: theme=red; Expires=Sat, 03 Feb 2024 01:33:50 GMT;
  Path=/
7 Vary: Cookie
8 Content-Length: 15482
9
10 <!DOCTYPE html>
11 <html lang="en">
12   <head>
13     <meta charset="utf-8">
14     <meta name="viewport" content="width=device-width,
     initial-scale=1, shrink-to-fit=no">
15     <title>
         Jeff's Sandbox
       </title>
```

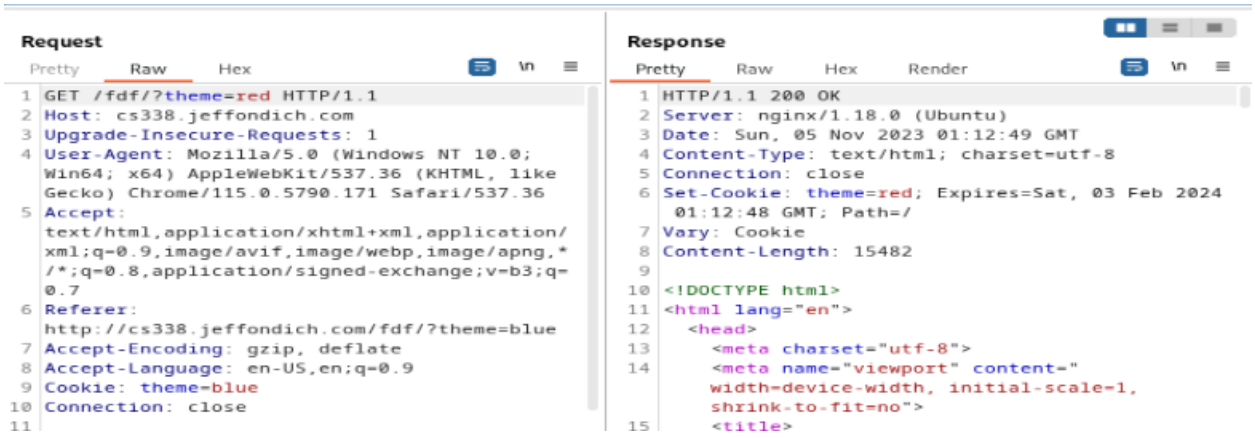
In the request, the cookie “theme” is set to “default”. Then, the response sets the cookie “theme” to “red”.

After pressing the button to change the theme to blue, I got the following request and response:



In the request, the cookie “theme” is set to “red”. Then, the response sets the cookie “theme” to “blue”.

After pressing the button to change the theme to red again, I got the following request and response:



In the request, the cookie “theme” is set to “blue”. Then, the response sets the cookie “theme” to “red”.

These cookie values match what I saw with the Inspector.

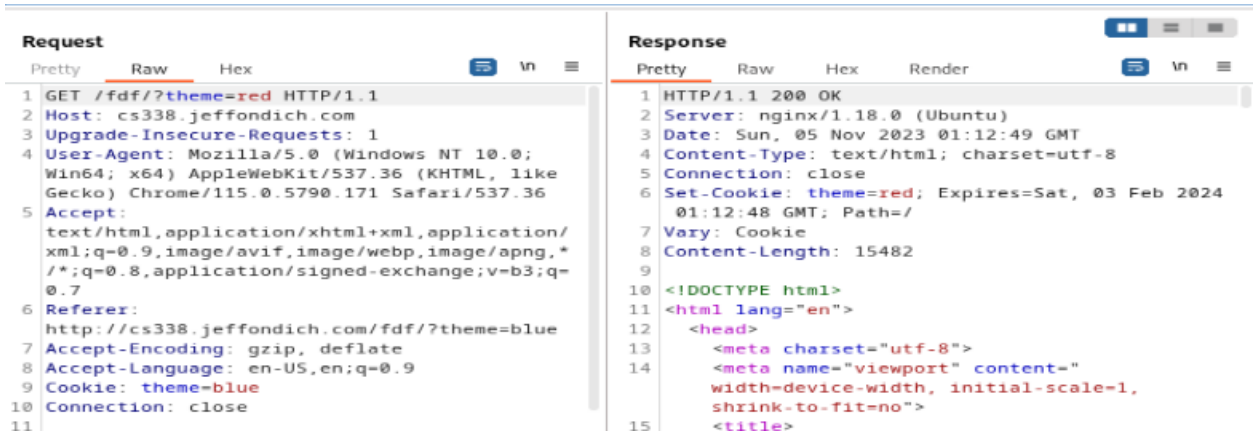
D. The red theme (the theme I last selected) is still selected.

E. When I quit my browser and relaunch it, the browser sends an HTTP GET request to the FDF server requesting /fdf/. In this HTTP GET request, there’s a header marked “Cookie:

theme=blue". The FDF server processes this cookie and changes the FDF's theme accordingly (the FDF server sends back the "blue-themed" html).

F. When I click the change theme button (going from blue to red, for example), my machine sends an HTTP GET request to the FDF server requesting /fdf/?theme=red.

Then, the FDF server responds by sending over /fdf/?theme=red and sets the cookie "theme" to "red".



G. I could go into my browser's Inspector and edit the FDF server's "theme" cookie, manually changing its value from "red" to "blue", for example. Then, I would reload the page and the theme would be changed.

H. Within Burpsuite's Proxy tool, if I have an HTTP GET request for /fdf/, I can manually edit the "theme" cookie and I'll get whatever FDF theme I entered.

If I have an HTTP GET request for /fdf/?theme=blue or /fdf/?theme=default and I want my FDF theme to be red, I could just change /fdf/?theme=blue or /fdf/?theme=default to /fdf/?theme=red. Then, my FDF theme will be red. I can do something similar when I want my FDF theme to be blue or default.

I. On Kali, I used the default Burpsuite browser. It stores its cookies in /home/kali/.BurpSuite/pre-wired-browser/Default/Cookies. This file looks like this:

�QLite format 3@ .f�

$�g�

 ��5Gindexcookies_unique_indexcookiesCREATE UNIQUE INDEX cookies_unique_index ON cookies(host_key, top_frame_site_key, name, path)�@WtablecookiescookiesCREATE TABLE cookies(creation_utc INTEGER NOT NULL,host_key TEXT NOT NULL,top_frame_site_key TEXT NOT NULL,name TEXT NOT NULL,value TEXT NOT NULL,encrypted_value BLOB NOT NULL,path TEXT NOT NULL,expires_utc INTEGER NOT NULL,is_secure INTEGER NOT NULL,is_httponly INTEGER NOT NULL,last_access_utc INTEGER NOT NULL,has_expires INTEGER NOT NULL,is_persistent INTEGER NOT NULL,priority INTEGER NOT NULL,samesite INTEGER NOT NULL,source_scheme INTEGER NOT NULL,source_port INTEGER NOT NULL,is_same_party INTEGER NOT NULL,last_update_utc INTEGER NOT NULL)f�/tablemetametaCREATE TABLE meta(key LONGVARCHAR NOT NULL UNIQUE PRIMARY KEY, ����;last_compatible_version18e_autoindex_meta_1meta

ersion18#mmap_status-1

����last_compatible_version

2�c5 ersi/h�珺
cs338.jeffondich.comthemev11d�▒|(�Z�@K�BP�//o,�k /h�3 �P/h�3�m

�� 5 cs338.jeffondich.comtheme/

I also used Firefox, which stores its cookies at /home/kali/.mozilla/firefox/169m5til.default-esr/cookies.sqlite. This file looks like this:

SQLite format 3@

}}OO##etablemoz_cookiesmoz_cookiesCREATE TABLE moz_cookies (id INTEGER PRIMARY KEY, originAttributes TEXT NOT NULLost TEXT, path TEXT, expiry INTEGER, lastAccessed INTEGER, creationTime INTEGER, isSecure INTEGER, isHttpOnly INTEGER, ameSite INTEGER DEFAULT 0, rawSameSite INTEGER DEFAULT 0, schemeMap INTEGER DEFAULT 0, CONSTRAINT moz_uniqueid UNIQUE (Endexsthemedefaultcs338.jeffondich.com/eies]W5|]W5|

5 themecs338.jeffondich.com/

Part Two - Cross-site Scripting (XSS):

A. Here is a step-by-step description of the nature and timing of Moriarty's attack on users of the FDF:
- Moriarty creates a post on the FDF server. This post contains some nefarious Javascript. Let's assume that this Javascript runs when the post is clicked on.
- At some point, a user of the FDF clicks on Moriarty's post. The Javascript executes and does something nefarious on the user's machine.

B. A bit of Javascript in one of these posts could make it so that, when a client clicks on the post, the Javascript gets the client's IP address (more details on how to do that here: https://www.geeksforgeeks.org/how-to-get-client-ip-address-using-javascript/). Then, with Javascript's fetch method, the client's IP address can be sent to the attacker's private server (fetch sends a network request to a URL. If the URL looked something like: http://attackerServer?IPAddr, it could pass the IP address of the client to the attacker's server). With the client's IP address, you can find their approximate location (for example, this site will tell you your approximate location based on IP address: https://www.iplocation.net/). With this location you could probably do something pretty nefarious (stalking the client, for example).

C. A bit of Javascript in one of these posts could make it so that, when you click on the post, you're redirected to a different website. While this could be relatively harmless (if you're rick-rolled for example), consider a scenario in which you're sent to a malicious website where, for example, it says you need to log-in with your Carleton username and password in order to read the posts on the FDF server. Because you trust the FDF server, you trust that this website it's routed to is legitimate and thus, you hand over your username and password.

D. The server can sanitize the input before writing it to a page. One way of doing this is by making a whitelist of user input values. Then, if you encounter a non-whitelisted value, you convert it to an HTML entity. For example, this might look like converting < to < and converting > into >.