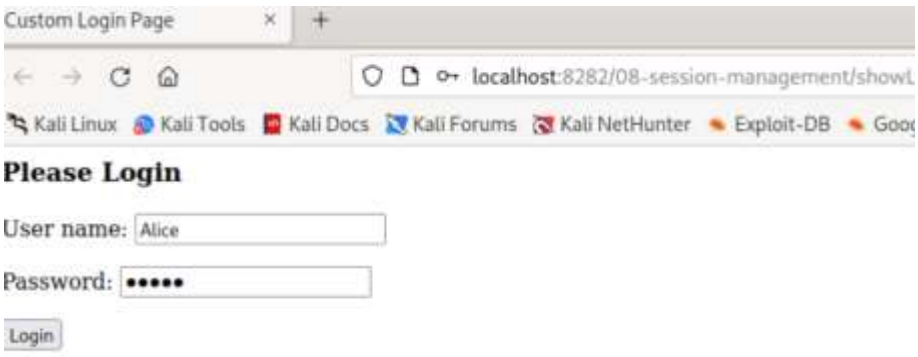
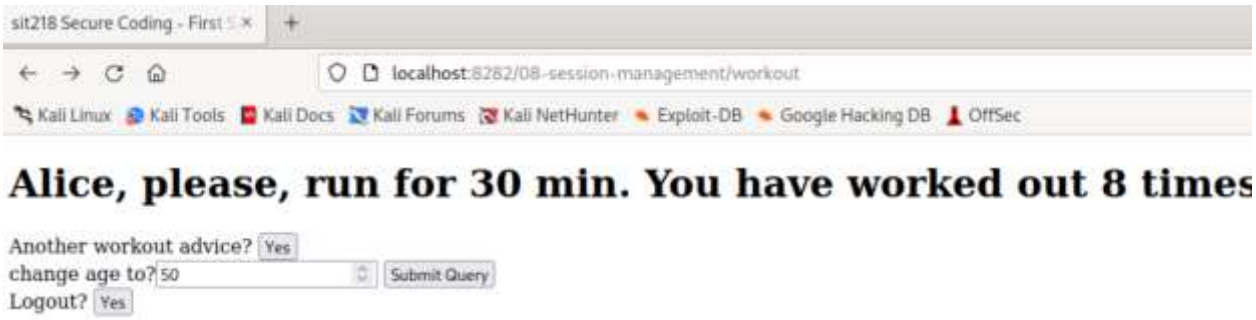
Task 7.1P: Session management and CSRF attacks

- To accomplish my task, I looked into the CSRF prevention technique in 08-session-management program used in the 08-session-management program. In this case, I paid particular attention to how this application was protecting it from CSRF attacks using the value of the HTTP Referrer header. I selected cross site request forgery and performed an attack then analyzed the outcome, especially concerning the interaction between the prevention logic and problems. Therefore, I first logged into the application with the ID, being 'Alice' so as to use it under normal circumstances. This is also possible for Alice to write fresh age to the server by using the GET request only.

```
                client.addWorkout();
                ClientDAO dao = new ClientDAOImpl();
                dao.updateClient(client);
        }
    }
    @RequestMapping("/change")
    public String changeAge(@RequestParam("age") int age,
            Model model, HttpServletRequest request) {
/*

        //logic to prevent CSRF attacks

        this.referer2 = request.getHeader("referer");
        if (this.referer1 != this.referer2)
        {
            HttpSession session = request.getSession(false);
            if (session != null) {
                session.invalidate();
            }
        }
*/
        //System.out.println(referer);
            ClientDAO dao = new ClientDAOImpl();


            HttpSession session = request.getSession();
            String uname = (String) session.getAttribute("user");
```

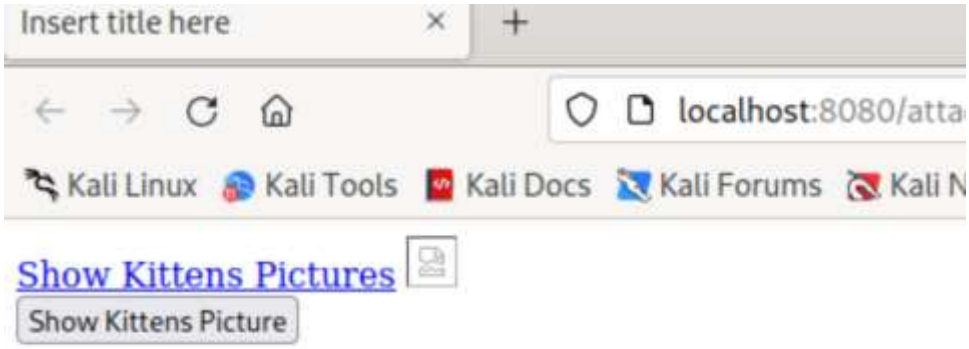


- Following that I set up an attacker website with a special GET request included for the process of the CSRF attack. This attacker website link or button can be clicked with possible result of changing Alice's Meanwhile they are signed in it updates their age in the application database.

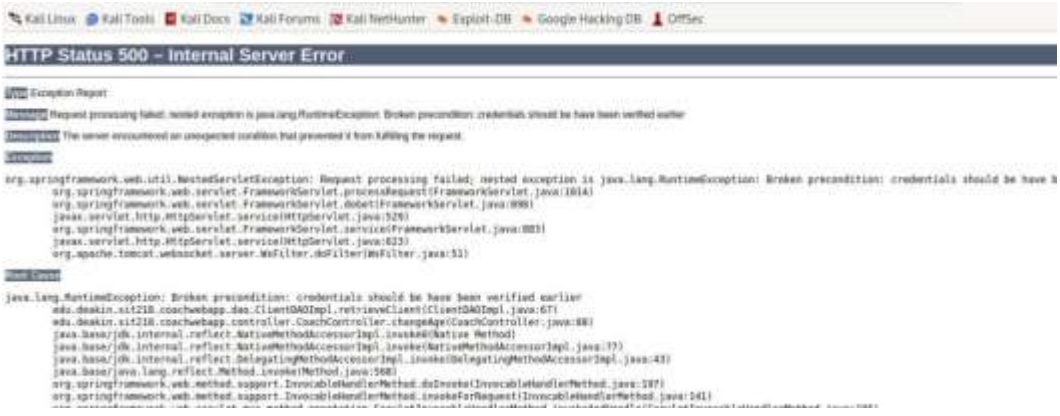


The website loads again when I click the "Yes" button in the "Another workout advice?" section.

- I was on the attacker page and monitored the HTTP requests through the Firefox developer's tools during the attack. The virus was inside the "Show Kittens Picture" button and I noticed that the application stopped the activity the moment the participant clicked on it. When filling the age field of the application instead of changing it to Alice's real age an error message was displayed.


Insert title here
localhost:8080/atta
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali N
Show Kittens Pictures
Show Kittens Picture


Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking
You age has been updated
Another workout advice? Yes
change age to? Submit Query
Logout? Yes


Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
HTTP Status 500 – Internal Server Error

Prevention Logic

The CSRF prevention method implemented in this application involves checking for the Referrer in the HTTP requests received. Since the computer wants to avoid accruing responsibility from a malicious request, the header providing information on where the request is coming from is checked. This tactic is effective because most CSRF attacks are from fake or third party web sites. Authorized modifications only are allowed; the software also ensures that the age of Alice, for example, can only be changed by the proper domain.

When This Mitigation Fails

- HTTPS to HTTP Requests
  - ✓ This is due when translating from HTTPS to HTTP referrer information may be scrubbed, thereby impinging on the ability of the program to run through the specific tests.
- Referrer Header Stripping
  - ✓ It is only possible in some situations when privacy is partially or fully a concern because of specific browsers or different security settings head stripping may occur in referrers. It has been observed sometimes the application is not able to provide a guarantee on the authenticity of the request that is made which in turn creates an opportunity for cross-site request forgery (CSRF) attacks.
- Same-Domain Attacks
  - ✓ There is a possibility that this mitigation may be reduced if the CSRF attack comes from the same domain or subdomain of the cookies in charge. In other words it means as it is represented there is no possibility of the application to be able to stop the particular attack yet the referrer is going to be authentic.

- For instance, the 08-session-management applies the HTTP referrer header to prevent CSRF attacks. Such approach has been proved quite efficient in preventing attacks from well distinct domains, let alone attacks from related or associated domains only. It isn't perfect, though. To eliminate referrer checking, there is one strategy that should be implemented, and they are, applying of anti-CSRF tokens. Since these tokens ensure that each request is authorized during the session of the user, there is an added security in the system.