CSE467 Homework #3: Web Security

Due: Nov. 24, 11:59 PM

- **Late submission policy.** Please refer to the course web page.
- **Submission guidelines.**
  - You should submit both your report and flags.
    - Solve the problems and submit the flags to our homework webpage: `http://10.20.12.187:4000`. This server can only be accessed from the UNIST internal network. Please use a VPN to access from outside.
    - You should upload a single PDF file on BlackBored. Your report must describe the answer to each question in this homework. Your report (`your_ID-last_name.pdf`) can be written in either English or Korean.
  - The name of the PDF file should have the following format: `your_ID-last_name.pdf`. For example, if your name is Gil-dong Hong, and your ID is 20231234, then you should submit a file named "`20231234-Hong.pdf`".
  - If your solution includes some code (Python, C, etc.), you should embed them in your PDF.
- **Capture The Flag (CTF) guidelines.**
  - You can find each problem on our homework webpage: `http://10.20.12.187:4000`
  - If you solve each challenge, you will be able to obtain a flag. Submit the found flag to the website. Each flag is in the following format: `flag{[0-9a-f{32}]}` (e.g., `flag{1a79a4d60de6718e8e5b326e338ae533}`)
  - Your score in the CTF scoreboard is nothing to do with the actual score for your homework. The CTF score is just for fun.
  - Do not attack the CTF environments, including web services!
  - If you think the services are not working properly or have any questions, please publicly upload your question on the BlackBored.
- **XSS attack guidelines.**
  - For XSS challenges (Problems 4–5), we prepared an imaginary victim and stored the flag in this victim's cookie. Therefore, you need to find a vulnerability in the web service, create a malicious URL, and send the URL to this user so that you can read the cookie.
  - Especially, you should manually send a malicious URL to this victim user via this webpage: `http://10.20.12.187:4004/check.php`. When you send the URL to the victim user, she will automatically visit the URL. This process might take at most 10 seconds.
  - To read the user's cookie, you may need your own server (Recall the `attacker.com` in pages 43 and 53 of the lecture slide "Client-side Web Security (1)" [1]). We recommend using the following service: `https://webhook.site/`. Through this service, you will get your unique URL that acts like `attacker.com`, and you will be able to see logs (e.g. query strings, form contents, etc.) of all accesses to that URL.

Problem 1. Login as admin (15 points)

In this challenge, you should sign in to the web service as admin without knowing the password. Once you successfully sign in to the service, you will find the flag.

(Hint) In the database, there exists a table with the following structure.

```
CREATE TABLE user (
    idx INTEGER,
    username TEXT,
    password TEXT
);
```

(a) (2 points) Describe the SQL query used in this web service

(b) (3 points) Describe the vulnerability in this web service.

(c) (8 points) Explain in detail how you can exploit this vulnerability to get the flag.

(d) (2 points) What is the flag?

Problem 2. Get admin's password (25 points)

In this web service, you need to find out the password of admin, which is the flag of this challenge.

(Hint 1) In the database, there exists two tables with the following structures.

```
CREATE TABLE user (  // Sample users: guest/guest, test/test, test2/test2, test3/test3,
    idx INTEGER,     //                abc/abc, abcd/abcd, abcde/abcde
    username TEXT,
    password TEXT
);
CREATE TABLE privileged_user (   // Admin's account is stored in this table
    idx INTEGER,
    username TEXT,
    password TEXT
);
```

(Hint 2) For your information, Section 6 offers instructions for crafting your exploit.

(Hint 3) A flag is in the following format: `flag{[0-9a-f{32}]}`

(a) (5 points) Describe the vulnerability in this web service.

(b) (1 points) What is the HTTP method of the request sent by the browser when you click the "Check!" button (Hint: Refer to the network tab in the browser developer console)?

(c) (1 points) What information is included in the body content (i.e., payload) of the HTTP request sent by the browser when you click the "Check!" button (Hint: Refer to the network tab in the browser developer console)?

(d) (13 points) Explain in detail how you can exploit this vulnerability to get the flag (admin's password). If you used a script to exploit it, please include the script in the writeup.

(e) (5 points) What is the flag?

Problem 3. Get color (15 points)

In this challenge, you need to read the contents of the file `/var/www/flag.txt`.

(a) (3 points) Describe the vulnerability in thie web service (Hint: How does the URL change as you select a color?).

(b) (10 points) Explain in detail how you can exploit this vulnerability to get the flag .

(c) (2 points) what is the flag?

Problem 4. XSS - Music search (20 points)

In this challenge, you need to read the victim's cookie!

(a) (5 points) Describe the vulnerability in this web service. You need to specify which type of XSS this vulnerability is.

(b) (11 points) Explain how you can exploit this vulnerability to get the flag.

(c) (4 points) What is the flag?

Problem 5. XSS - Service center (25 points)

In this challenge, you need to read the victim's cookie!

(a) (5 points) Describe the vulnerability in this web service. You need to specify which type of XSS this vulnerability is.

(b) (15 points) Explain how you can exploit this vulnerability to get the flag.

(c) (5 points) What is the flag?

6 Appendix

Crafting Exploits using Python3. It is recommended to use a programming language like Python to create your own exploit. Writing one line of code is more efficient than manually sending a request 400 times. We provide guidelines for crafting exploits using `Python3`.

```
import requests

url = "http://10.20.12.187:4002/"      # Target URL
data = "key": "value"                  # Body content (payload) of your request

req = requests.post(url, data = data)  # Send your (post) request to the target URL
print (req.content)                    # Get the response (website content)
```

References

[1] CSE467: Computer Security. 2023. 14. Client-side Web Security (1). https://websec-lab.github.io/courses/2023f-cse467/slides/lecture15-client-side.pdf.