Elinor Holt

Prof. Goldberg

CS 357

Nov 1, 2024

Challenge 5: Salted SHA256

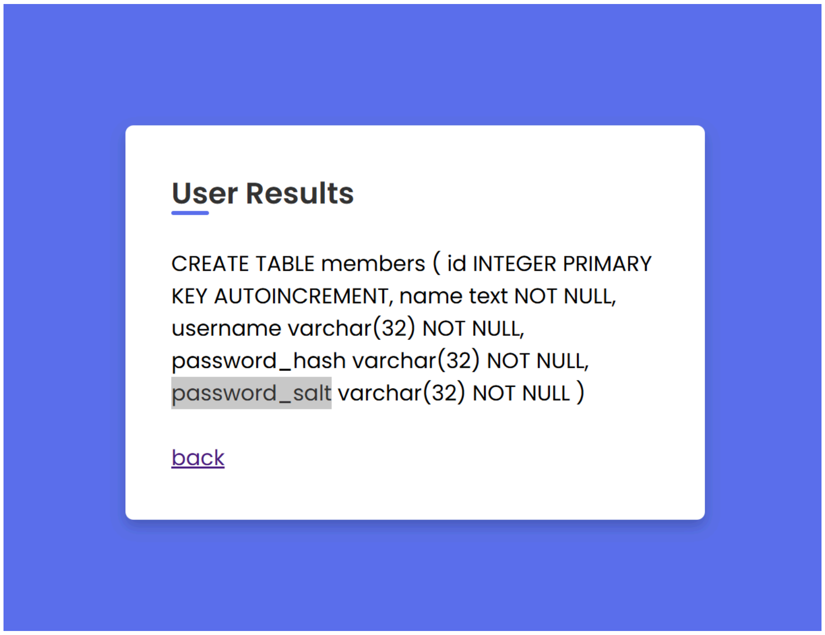
I. Recon:

A. I used the same methods from Challenge 3 to find the table and column names in the database.

B. The main difference here was locating the column that stores the unique salts associated with each member’s password hash since the database has been altered from Challenge 4.

C. Using the same method, the output is different now that the “**password_salt**” column has been added.

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

D. 

User Results

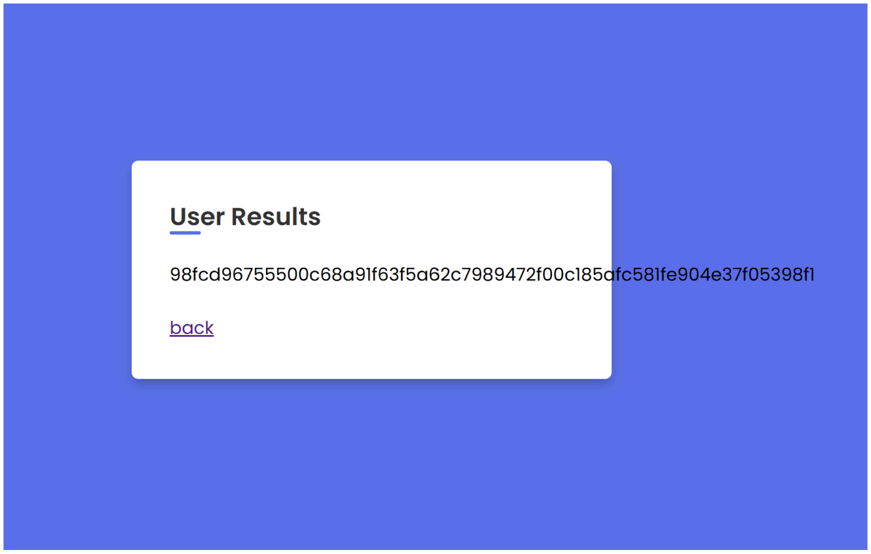
CREATE TABLE members (id INTEGER PRIMARY KEY AUTOINCREMENT, name text NOT NULL, username varchar(32) NOT NULL, password_hash varchar(32) NOT NULL, password_salt varchar(32) NOT NULL)

II. Exploit Instructions (Automated in code):

A. I reused the exploit method from the search page to find the password hash associated with the admin account by submitting the following SQL query:

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

B.



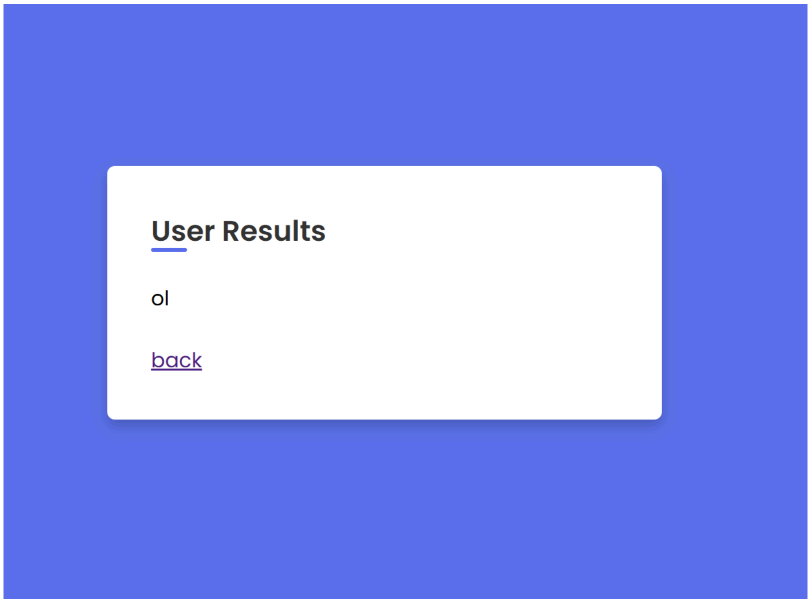
User Results

98fcd96755500c68a91f63f5a62c7989472f00c185afc581fe904e37f05398f1

C. This query retrieves the password_hash for the member with the username “admin.” Starting the query with a unique string (not found in any member names) simplifies the output, so the results display only the password_hash. The SHA256 hashing algorithm now produces a longer, 64-character hash.

D. I used the same method to find the password_salt associated with the admin account:

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

E. 

User Results

ol

F. As expected, the salt is a random lowercase two-letter string.

G. Knowing the salt and password hash associated with the admin account lets us replicate the original hashing process. I tried hashing potential passwords until one generated a hash which matched the admin's.

H. We are told the password is a five digit numerical value, so my Python script generates possible passwords with randomly selected five digit long strings of numbers. Then the program will iterate through the possible passwords and hash each one until it finds a hash which matches the admin's password hash and prints it out. This password can be used to log into the admin account.

III. References:

A. https://www.geeksforgeeks.org/python-generate-random-string-of-given-length/

B. https://docs.python.org/3/library/hashlib.html (I used this library to automate the hashing function)