Instructions:

- Submit typed solutions, following the rules in the syllabus.
- Include the “disclaimer” given in the syllabus!
- “Describe an attack” means show the final result (in this case, a maliciously crafted URL) and also describe what it does, why it does it, and your thought process of how you found it (as if you were reporting a vulnerability to a software author).

This homework refers to the login form here:

http://web.engr.oregonstate.edu/cgi-bin/cgiwrap/rosulekm/sqli-hw.cgi

You can check out the source code here:

http://web.engr.oregonstate.edu/cgi-bin/cgiwrap/rosulekm/highlight.cgi?sqli-hw.cgi

As you can see, this script uses an SQLite database containing the user data.

Note: You can play around with SQLite on the engr servers by typing sqlite3.

1. From looking at the source code, what can you determine about the schema of the database?

2. Looking at the source code, identify and describe exactly why the code is vulnerable to SQL injection.

3. Does any information from the database get included in the visible output (if so, where)?

4. Describe an SQL injection attack that allows you to determine how many columns are in the users table.

   Important Hint: SQLite supports the UNION operator to return the union of two SELECT statements. You should make the script execute an appropriate UNION SQL statement, then follow the error messages.

Note: Besides the table that holds user login data, every SQLite database has a builtin table called sqlite_master with the following schema:

```sql
CREATE TABLE sqlite_master (
    type TEXT,
    name TEXT,
    tbl_name TEXT,
    rootpage INTEGER,
    sql TEXT
);
```
This table contains information about all of the other tables in the system. You can \texttt{SELECT} from this table just like any other. I encourage you to try it on your own local SQLite database.

5. Although we already know that the table is named \texttt{users}, describe an attack by which we could have used SQL injection to find out the name of the table without having access to the source code of the script (but knowing that it used SQLite).

6. Describe an attack that allows you to learn the SQL schema of the \texttt{users} table. (table schema = SQL statement used to create the table)

7. Describe an attack that allows you to learn the entire contents of the \texttt{users} table!
   
   \textit{Hint}: The script executes an SQL statement and only looks at a single row of the result. So you'll have to extract the database piece-by-piece. You might find the following idea useful:

   \begin{verbatim}
   SELECT MIN(x) FROM sometable  --> returns x="alpha"
   SELECT MIN(x) FROM sometable WHERE x > "alpha"  --> returns x="beta"
   ...
   SELECT MIN(x) FROM sometable WHERE x > "zeta"  --> [empty results]
   \end{verbatim}

8. Suggest a patch to the code, to fix the SQL injection vulnerability.