CSCI 3330 Assignment 3

Deadline: October 2, 2025, 10:00pm

1 Background

In this assignment, we will be further exploring Unix permissions, SQL, and SQL injection vulnerabilities. Specifically, by completing this assignment, you will:

- Understand in greater depth how operating systems process permissions and, more generally, paths.
- Explore additional ways of curating rows returned by SQL queries.
- Demonstrate how ineffective string sanitization leads to SQL injection and craft payloads that work under progressively stronger (but still flawed) defenses.

2 AI policy

You may use AI to learn background concepts. You may not use AI to generate, verify, or iterate on solutions for this assignment.

Allowed:

- Concept definitions and explanations: "What does the `chmod` command do?", "What is sha256?", "What does `replaceAll()` do?"
- Generic examples not derived from this assignment: "Walk through an example of a `SELECT` query using the `ORDER BY` clause," "Walk through an example of an SQL injection attack."
- Non-generative tooling: man pages, official documentation, calculators, programming languages, and trying something for yourself.

Handle with care:

- Tools which improve grammar or clarity.
- IDE autocompletion or documentation lookup that does not inject solution content.

Not allowed:

- Any hint, outline, or solution for these specific questions (including rephrased or "similar" versions), e.g., "Help me with the attack in Q2," "Write an SQL injection attack to log me in as dbadmin," or "Check my answers."
- Pasting assignment text or numbers into AI to tailor examples.
- Using AI to generate any text, code, calculations, or any other artifacts that you submit.

Ask me questions instead! When in doubt, don't use AI, because **you won't have access to it on tests**, for which these assignments are your preparation.

3 Questions and SQL warmup

1. Suppose your current working directory contains two directories "foo" and "baz", that "foo" contains a subdirectory "bar", and that you run the following command to read the file `bletch.txt`:

   ```
   cat foo/bar/../../baz/bletch.txt
   ```

   (a) Which directories do you need to be able to traverse (`x`) to successfully run this command?

   (b) Which files (including directories) do you need to be able to read (`r`) to successfully run this command?

   (c) What does this tell us about how the operating system processes paths?

   Hint: Try it out using `mkdir` and `chmod`.

2. As we have discussed in class, SQL allows filtering by conditions. However, we may combine multiple conditions using `AND` and `OR` operators. As in most languages, `AND` has higher precedence than `OR`. (The intuition for this is that taking the *and* of multiple bits is similar to taking their product (`*`) and taking the *or* of multiple bits is similar to taking their sum (`+`). Parentheses can be used to force the `AND`/`OR` operators to apply in a different order. Which of the following SQL statements are equivalent?

   (a) `SELECT * FROM t WHERE x = 0 OR y = 1 AND z = 2`

   (b) `SELECT * FROM t WHERE x = 0 OR (y = 1 AND z = 2)`

   (c) `SELECT * FROM t WHERE (x = 0 OR y = 1) AND z = 2`

3. This question concerns the SQL tables in the environment here. Write a statement that selects all UFO reports whose id is greater than 1001 and either the credibility score is greater than 3 or the case status is "open".

4 SQL login challenge

For the questions in this section, you will be attacking an SQL login page under various conditions of increasing difficulty. You may ***NOT*** use the `UNION` keyword in this section.

1. The login page for this question is available here. Use SQL injection to successfully log in. The code in JavaScript builds the query as follows:

   ```
   return `SELECT id, username, password, role FROM users
   WHERE username = '` + u + `' AND password = '` + p + `' AND role <> 'disabled'`;
   ```

   What strings did you use for *username* and *password* to successfully log in? For this problem, you may log in as any user and may not assume any knowledge about the names of accounts in the database.

2. The login page for this question is available here. Use SQL injection to successfully log in. The code in JavaScript builds the query as follows:

   ```
   return `SELECT id, username, password, role
   FROM users
   WHERE username = '` + u + `' AND
   password = '` + p + `' AND
   role <> 'disabled'`;
   ```

   What strings did you use for *username* and *password* to successfully log in? For this problem, you may log in as any user and may not assume any knowledge about the names of accounts in the database.

3. The login page for this question is available here. Use SQL injection to successfully log in. The code in JavaScript builds the query as follows:

   ```
   return `SELECT id, username, password, role
   FROM users
   WHERE username = '` + u + `' AND
   password = '` + sha256(p) + `' AND
   role <> 'disabled'`;
   ```

   What strings did you use for *username* and *password* to successfully log in? For this problem, you may log in as any user and may not assume any knowledge about the names of accounts in the database.

4. Using the same login page as the previous problem, now use SQL injection to successfully log in as the user `dbadmin`. What strings did you use for *username* and *password* to successfully log in as the user `dbadmin`?

5. The login page for this question is available here. Use SQL injection to successfully log in as the user `dbadmin`. The code in JavaScript builds the query as follows:

```
return `SELECT id, username, password, role
FROM users
WHERE username = '` + escapeSingleQuotes(u) + `' AND
password = '` + sha256(p) + `' AND
role <> 'disabled'`;
```

where `escapeSingleQuotes()` is defined as

```
function escapeSingleQuotes(s) {
    return s.replaceAll("'", "\\'");
}
```

What strings did you use for *username* and *password* to successfully log in as the user `dbadmin`?

6. The login page for this question is available here. Use SQL injection to successfully log in as the user `dbadmin`. The code in JavaScript builds the query as follows:

```
return `SELECT id, username, password, role
FROM users
WHERE username = '` + escapeQuotes(u) + `' AND
password = '` + escapeQuotes(p) + `' AND
role <> 'disabled'`;
```

where `escapeQuotes()` is defined as

```
function escapeQuotes(s) {
    return s.replaceAll("'", "\\'").replaceAll('"', '\\"');
}
```

What strings did you use for *username* and *password* to successfully log in as the user `dbadmin`?

5 SQL exfiltration challenge

For the questions in this section, you will be exfiltrating data from an SQL login page under various conditions of increasing difficulty. You ***may*** use the `UNION` keyword in this section.

1. Using the page in the previous section's Question 1, use SQL injection to exfiltrate the password of a user. What strings did you use for *username* and *password* to successfully exfiltrate the password of a user? Whose password did you exfiltrate and what was the password?

2. Using the page in the previous section's Question 6, use SQL injection to exfiltrate the password of the `dbadmin` user. What strings did you use for *username* and *password* to successfully exfiltrate the password of the user `dbadmin`? What was the password?

6 Submission

Email your answers in a single `*.txt` or `*.pdf` file attachment to my email address in the Syllabus.