# Mini Project 4: SQL Injection
(due May 4, 2023)

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when the user's inputs are not correctly checked within the web applications before being sent to the back-end database servers. The SQL injection attack is one of the most common attacks on web applications.

In this mini-project, we have created a web application that is vulnerable to SQL injection attacks. Please use it as a testbed to find ways to exploit the SQL injection vulnerabilities and demonstrate the damage that can be achieved by the attack.

## Important Note – during your attack, please DO NOT:

- Delete any record.
- Access (read from or write to) other tables or databases.
- Alter table structure.
- Insert HTML code or JavaScript.
- Insert Insert a large number of records (just add a few to show the attack works).
- Try to execute shell commands.

## Setting

You can access the vulnerable web application at:
`https://cae.ittc.ku.edu/sqli/login.html`

The backend of the website is a MySQL database with a table named "users".

- The schema is: users(first, uname, passwd, profile)
- The default password for each user is the same as the username.
- The table only stores hashed passwords. It uses the PASSWORD() function provided by MySQL to create a password hash. Please refer to the below link to see how to call the password function to create hashed passwords: https://dev.mysql.com/doc/refman/5.6/en/password-hashing.html.

## Tasks

Please compose malicious input to exploit this vulnerable web application and achieve the following attack goals:

1. Impersonate an existing user without providing the password. This means the attacker can log in as a valid user in the database (e.g., the username is jacob).

2. Steal all records in the table. This is to pretend you have no information about any valid user at all.

3. Insert ONE new record to the table about yourself (use your name/nickname/screen name to let us know the record is inserted by you). Log in as the new user and take a screenshot. **Hint:** The current setting allows multiple queries concatenated by a semicolon. You will get a "login failed" page, but it does not mean your attack fails. You can check the database (e.g., run task 2 again) to see if your record is inserted successfully.

4. Try the above attacks in the safe web application at: `https://cae.ittc.ku.edu/sql/login.html` and report if they can succeed or not.

**Potential issues with Task 3:** If you get a "connection was reset" error when you attempt to include multiple queries in your input, it is highly likely caused by your local anti-virus software or firewall (such as Kaspersky). If this happens to you, here is a list of things that you can try:

- You can simply pause the anti-virus software for a while, and then your input may go through.
- You can add a space right before the semicolon (this is the trick to avoid the issue in most anti-virus systems).
- If it still doesn't work, you can use KU's virtual desktop at `https://technology.ku.edu/services/virtual-lab`. Note that this system also has a firewall that will block your SQL injection attempt, so you still need to add a space before the semicolon to make it work.
- If you have tried all the above options and still get the error, you can send your query string to me or the TA and we will execute it for you.

## Submission

Please submit a short report to describe what you have done for each task and include screenshots of your results in your report.