Assignment 5 - Security

Format of your written assignment hand-in

A pdf-document of up to four pages.

Firstly, please list the members of the group that have cooperated on the assignment.

Then write your answers to the assignment questions beginning with Question 1, 2, etc. Answers that are programs should be shown directly as text (not indrectly as links). Answers that are screenshots (images) should be merged into the pdf-file.

Deadline of hand-in

Please upload your pdf-document to Moodle under "Assignment 5" no later than November 11 at 23:55. Please have one goup member submit a single solution for the whole group. Group cooperation is encouraged. I also encourage each group member to work individually on the assigment questions, in addition to working jointly in your group.

Complex IT Systems - Fall 2024

Questions 1-3 are about SQL Injection.

Question 1. Define a stored database function that returns a table of courses from the course table in the university database. You may name the stored function safe_course(). The function should have one input parameter of type VARCHAR(8). The function should return courses whose course id match the input. (There will be at most one such course, since the course id is a primary key of the table). Naturally, the function should leave out any course offered by the Biology department. Show the SQL code that defines the function.

Question 2. In SQL-Injection-Frontend, in the switch statement of Program.cs, define an option 'sc' that calls a method safeComposedQuery(). Also, in QueryConstructor.cs, define method safeComposedQuery(). The new function should be an improvement over composedQuery() in two ways: firstly, it must call the stored database function safe_course(); and secondly, it must use separate parameter passing, including by calling query/3 as defined in PostgreSQL_client.cs.

Question 3. Define an SQL injection attack that works when option 'c' is selected, but fails when option 'sc' is selected. Provide a screenshot of the succesfull attack and a screenshot of the failed attack.

Questions 4 - 6 are about passwords.

Question 5. Implement a check that the password provided by a new user at registration contains more than eight characters. Also check that the password does not contain the username. Obviously, the password 'admindnc' for username 'admin' will fail this test. Checks should be done in the definition of method passwordIsOK() in Authenticator.cs.
(Perhaps you wonder why passwordIsOK() is defined as a virtual procedure. This allows for implementing modifications to the method in a subclass of class Authenticator. This helps me maintain program variants with and without modifications. You are free to hardcode you modifications directly into class Autenticator).

Question 5. Implement iterative hashing. Hint: In the C# source file Hashing.cs, in the definition of method hashSHA256(), you may modify the second parameter in the call to function iteratedSha256(). What number of iterations appears to be reasonable on your computer?

Question 6 is about passwords *and* SQL injection.

Question 6. In Authenticator.cs, the method sqlSetUserRecord() defines a string that is an SQL command. The SQL command is then used in method register(). Is the method vulnerable to SQL injection?