# Security Practical 3

Dr Chris G. Willcocks

Last Modified: November 20, 2017

## Practical 3

### Background

Welcome to the third practical. In this practical we will be implementing what we learnt in the lecture on databases and extending our web server accordingly. We will be using a SQL database called H2 primarily for its simplicity; however I recommend a different DBMS such as PostgreSQL or MongoDB in real applications, as these are generally better supported. We will be doing injection and inference attacks. This will also be a good opportunity for those already familiar with SQL to get some additional practice.

### Tasks

1. It is important that you are familiar with the SQL syntax in order to understand some of the more obscure queries/inference attacks.

   (a) **Take your time** to complete the SQL tutorial at: `https://www.w3schools.com/sql/sql_intro.asp` (you have to click 'Next' at each stage).

   (b) At the end of the tutorial you will be given a SQL quiz. Complete this. You are expected to score above 20 on this quiz.

2. Change directory to '6' then run the database server locally:

   (a) Change directory to 'H2/bin'

   (b) Run either h2.sh (UNIX) or h2.bat (Windows). UNIX users may need to chmod a+x the script.

   (c) A window should appear, telling you to open the database browser interface.

   **Listing 1: Setup**

   ```
   Driver Class:  org.h2.Driver
   JDBC URL:      jdbc:h2:~/test
   User Name:     user
   Password:
   ```

   (d) Enter the above information and click 'Connect'.

   (e) This will create a new database locally on your filesystem.

   (f) Locate the 'test' database file(s). If you mess things up, you can delete these and start over.

   (g) **Save a backup history of all your SQL statements in the following sections.**

   (h) Create a 'Users' table with the following data, where the ID field is the primary key and id and username have the NOT NULL constraint, whereas the other columns can accept NULL values. Grades and fines are floating point numbers. Make all text NVARCHAR(80) for now.

| ID | Username | Firstname | Lastname | Grade | College | Fines |
|---|---|---|---|---|---|---|
| 1 | jess | Jess | Adams | 86.4 | Castle | 4.21 |
| 2 | greg7 | Greg | Courier | 62.0 | Chads | 0.0 |
| 3 | alice4 | Alice | Smith | 57.9 | Castle | 1.21 |
| 4 | chrzi | Chris | Jackson | 73.8 | Cuths | 5.33 |
| 5 | lauran3 | Laura | Walker | 21.2 | Ustinov | 0.34 |
| 6 | ai219 | Andrea | Ivanov | 84.2 | Ustinov | 0.92 |
| 7 | seb123 | Seb | Elbert | 17.3 | Chads | 0.0 |

(i) Also, create a 'Private' table with the following data, where ID is a primary key and UserID is a foreign key. Store the wallets as NVARCHAR(80):

| ID | UserID | Wallet |
|---|---|---|
| 1 | 3 | KworuAjAtnxPhZARLzAadg9WTVKjY4kckS8pw38JrD33CeVYUuDm |
| 2 | 1 | Kwi5LPxVehUieD18AXiXTay9UDkRC7wLShe4tR5kzym1k2NhzEQ6 |
| 3 | 5 | L4mGG15YacXWPU7LHM8Lj2LboxabRriZGHFZb5eLDN7mPXPPAHQF |

(j) Do an inner join on the 'Users' and 'Private' table data, such that the query result looks like:

| Firstname | Lastname | Wallet |
|---|---|---|
| Alice | Smith | KworuAjAtnxPhZARLzAadg9WTVKjY4kckS8pw38JrD33CeVYUuDm |
| Jess | Adams | Kwi5LPxVehUieD18AXiXTay9UDkRC7wLShe4tR5kzym1k2NhzEQ6 |
| Laura | Walker | L4mGG15YacXWPU7LHM8Lj2LboxabRriZGHFZb5eLDN7mPXPPAHQF |

(k) You may wish to try using 'null' in the INSERT statements for the ID to auto-increment.

(l) Check the data is correct, e.g.: SELECT * FROM Users;

(m) Save your statements for creating the tables somewhere for later.

(n) Now disconnect (press the red icon in the upper-left).

3. Open a new terminal and compile and run Server.java, which needs h2*.jar in the classpath:

(a) Change directory to '6' then (if using UNIX):

**Listing 2: Setup**

```
javac -cp H2/bin/h2-1.4.196.jar:. Server.java && java -cp H2/bin/h2-1.4.196.jar:. Server
```

(b) Or (if using Windows):

**Listing 3: Setup**

```
javac -cp H2/bin/h2-1.4.196.jar;. Server.java && java -cp H2/bin/h2-1.4.196.jar;. Server
```

(c) If you get an error 'Database may already in use', make sure that you have indeed disconnected by clicking the red icon as mentioned in the previous steps.

4. SQL injection attacks:

(a) Open the website and do an SQL injection attack on the username input field to get all the 'Users' data.

i. Remember '−−' is a comment.

ii. You may wish to study the source code of Server.java

(b) Now do a more complex SQL injection attack on the username input field:

i. Find a way to display the private wallet information associated with the three users who have them.

ii. You may wish to copy information into their last name, or create new users accordingly.

5. Prepared statements:

(a) Reset the tables to how they were setup in question 2, if you have modified them when answering the above question.

(b) Fix the server from SQL injection attacks by using prepared statements.

i. H2 specific guide: `http://www.h2database.com/html/advanced.html#sql_injection`

(c) Try to do an SQL injection attack again on the username field to confirm that your prepared statements work as expected. Also confirm that you can still 'login' as normal users.

6. Inference attacks:

   (a) The instructor decides it would be helpful to put some class statistics on the website. The university has a policy in place where students should not be able to view the grades of other students. In another part of the website the instructor has a list of students by their college.

   (b) Can you infer the grades of each student based on the two views of the data?

      i. Just by using the website, what are the grades of Alice, Laura, and Seb?

   (c) The instructor still wants to display a list of anonymous grades for each student, but has asked you to make the list anonymous so you can't work out which grade corresponds to which student.

      i. Alter the SQL query to make the list of grades anonymous.

   (d) Write SQL queries in Server.java to return the average class grade and fines by college accordingly.

      i. Confirm the website returns the following:

         A. Average class grade:

            57.542857142857144

         B. Fines by college:

| | |
|---|---|
| Ustinov | 1.26 |
| Chads | 0.0 |
| Cuths | 5.33 |
| Castle | 5.42 |

   (e) The university has a policy in place where students should not be able to view the fines of other students.

      i. Login as Alice (her username is alice4).

      ii. What are the fines of Greg, Seb, Chris, and Jess?

   (f) What are your options for protecting against this?

7. This concludes the third practical. If you finish early, you may wish to extend the server to handle authenticated login with hashed and salted passwords stored in the SQL database.

---