# COM-402 exercises 2023, session 4: Data Security

## Exercise 4.1

Your application uses an SQL database which stores the names, grades and year of graduation of students.

- What mechanism can you apply to allow user Alice to only read data of students that graduate in 2025?

## Exercise 4.2

Consider an application that contains medical data. For more security, it uses two different tables: one with the personal data of patients and the other one with their medical conditions. Three database users are defined:

1. a user that can only read and write the personal data table
2. a user that can read the personal data and the medical data
3. a user that can read and write the medical data

The first user is used when patients update there personal data. The second user is used when the patients want to see their medical information. Finally, the third user is used when a doctor logs in to update the medical data.

- Which part of the application would have to be vulnerable to SQL injections, to allow a patient to read the medical information of another patient?
- What is a typical way of preventing an SQL injection?

## Exercise 4.3

- Why is transparent data encryption (the DB encrypts before writing to files) better than an encrypted file system (the OS encrypts the content of the files)?

## Exercise 4.4

- Give two reasons why it is important to salt password hashes.

## Exercise 4.5

You just built a very nice rainbow table that can crack 99% of 8 letter passwords. Compared to a brute-force attack, it uses 1,000 times less hash operations to find a password.

You are given a list of ten thousand hashes that need to be cracked.

- Is it going to be faster to crack the passwords with the rainbow table or with a classical brute-force attack?

## Exercise 4.6

- Why calculating a Windows password hash (based on MD4) is about 200,000 times faster than calculating a Linux password hash based on SHA512?

## Exercise 4.7

- Why is a graphics card that can calculate 10,000 hashes in parallel, not efficient for cracking password hashes like Argon or Scrypt?

# Solutions to the Exercises

## Solution 4.1

You can define a VIEW that only returns lines that contain the graduation year 2025. Then you can GRANT a privilege to user Alice to read from this view.

(The exact commands are:

```
CREATE VIEW Year_2025 AS SELECT * FROM com402.students
WHERE graduation=2025;

grant SELECT ON Year_2025 to alice@localhost;
```

)

## Solution 4.2

The part of the application that lets a patient see their medical data uses a database user that has access to the complete table of medical information. If this part of the application is vulnerable to SQL injection, then a patient could modify an SQL request to show other patients' medical data.

Prepared statements are a typical technique to avoid SQL injection vulnerabilities.

## Solution 4.3

In the first case, the data can only be seen by database users that have sufficient privileges or by administrators of the server that can dump the memory of the database. In the second case, users of the operating system that have the right to access the database files can also see the data.

## Solution 4.4

A different salt per hash forces the attacker to crack each hash separately. They can not try to crack several hashes with a single hash calculation. A random salt prevents the attacker from calculating the hashes in advance.

## Solution 4.5

Although a rainbow table reduces the effort needed to crack a single password, it can not crack multiple passwords at a time. You have to search for the corresponding end of chain of each hash to crack. So cracking 10,000 passwords will be 10,000 times longer that cracking one password. If you have been able to build a rainbow table, this means that the passwords are not salted. Thus, a brute-force attack can hash all possible passwords only once and search for the corresponding hashes in the list of 10,000 hashes. So even if brute-force is 1,000 times slower, it will still be 10 times faster than 10,000 rainbow table runs.

## Solution 4.6

SHA512 is indeed more complicated to calculate than MD4 but the main part of the difference is due to the fact a Linux hash is calculated with thousands of iterations (typically 5,000). This very efficiently slows down the cracking process.

## Solution 4.7

These hash functions are *memory hard*, which means that they need a certain amount of memory to be calculated efficiently. Although graphics card have many processing cores, they don't have enough memory per core to calculate the hash efficiently.