SLH 23-24
Lab #1

- This lab will be graded.
- You can answer in English or in French.
- The lab has to be done **individually**.
- **Destructive** behaviour (rm, DoS, ...) are **forbidden** and will be strongly penalised.
- Tools like **sqlmap** are not needed and are not allowed.
- The websites are only accessible with **the school's VPN**.

Description

The objective of this lab is to study various Top25 CWE attacks. You have to **attack** the vulnerable applications and **answer to the questions** in a report.

1 Website 1

Last year's students were tasked with improving the security of a grade management system named KING (KING Is Not GAPS). One of these students' project ended up in production, but the content of its database got leaked by an unknown hacker! The hacker generously made the email addresses and passwords of the students publicly available on `http://10.190.133.22:9002`.

1.1. What informations can you gather from the frontpage alone? How does the website function?

1.2. What is the IP of the databases containing the leaked logins? What information can you infer from them regarding their location? Give as much details as possible.

1.3. The hacker has also a private database storing a secret flag. What is this flag?

1.4. How did you find the flag? Explain your attack clearly.

1.5. What is the CWE you exploited? Cite at least the main one.

2 Website 2

You are given access to a cool website (`http://10.190.133.22:9004/`) that allows you to convert an image to black and white and change its brightness. The comment field is supposed to add the content in the metadata (the developer didn't do it, no time). The backend is written in Java and you came across a version of the source code that runs in a docker (it might not run locally without some modifications). You will find this source code on Cyberlearn in the folder `challenge2`. We give now a quick overview of the tools used to deploy this project:

Springboot

Java Spring Boot is a powerful framework that simplifies the process of building robust and scalable web applications. At the heart of a Spring Boot application lies the concept of controllers, which plays a pivotal role in handling incoming HTTP requests, processing them, and providing appropriate responses.

What is a controller? In Spring Boot, a controller is a component responsible for managing the flow of an application and handling HTTP requests from clients, such as web browsers, mobile apps, or other services. Controllers act as intermediaries between the user's requests and the application's business logic, facilitating the interaction between the client and the server. Let's analyze the following controller from the challenge:

```
@PostMapping("/secret")
public String decryptData(@RequestParam String secret) {
    try {
        String decryptedData = CryptoUtils.decrypt(secretData, secret);
        model.addAttribute("decryptedData", decryptedData);
        return "secret-result"; // Return a Thymeleaf template to display the
    decrypted data
    } catch (Exception e) {
        model.addAttribute("error", "Error decrypting data");
        return "secret-form"; // Return to the form with an error message
    }
}
```

`@PostMapping("/secret")` is an annotation that allow to create an HTTP endpoint which accept POST request. Thus, the following request would accept any POST request where the destination is `http://IP_SERVER/secret`
`public String decryptData(@RequestParam String secret, Model model) { ... }` is the function which intercept the request. In this example, `@RequestParam` allows to specify a `POST` parameter which should be given with the request. The `Model` parameter allow to specify a attribute for the templating engines, and thus create specific rendering based on the datas.
`return "secret-result";` allow to render a specific template. A template is a dynamic html page where we can specify where the model informations are set.

Application properties. `application.properties` is a configuration file commonly used in Java Spring Boot applications. It plays a significant role in configuring and customizing the behavior of a Spring Boot application. This file is typically located in the `src/main/resources` directory of a Spring Boot project.

Project dependencies

`pom.xml` is an essential configuration file used in Apache Maven, a popular build automation and project management tool for Java applications. The `pom.xml` file serves as the heart of your Maven project, defining various aspects of your project's structure, dependencies, and build process
You will find in the archive the configuration file of this challenge, and use it to identify the multiple dependencies of the project.

Tasks and Questions

2.1. One of the dependencies suffers from various CVEs, one of which being very critical. Provide the exact CVE number and the related CWE. Your IDE can help you finding vulnerabilities in dependencies. Alternatively, you can search manually.

2.2. Exploit the vulnerability. For this you will need:

- to find the location in the code of the CVE.
- to exploit it (you can search for resources online).
- Beware of the regular expressions used by the backend to filter some requests.
- The flag is encrypted but you can use the /secret endpoint to decrypt it once you exploited the CVE.

Provide in your report **an explanation of the attack** and the **flag**.

2.3. Which CWEs did you exploit to obtain the flag?

3 Website 3

The third website can be found at the url `http://10.190.133.22:9003/`. It allows to upload and download files on a server. Only **small** txt, jpg, jpeg, png, and pdf files can be uploaded. You will also find the Python code of the application on Cyberlearn in the folder `challenge3`.

3.1. Recover the `secret.txt` file. It can be found at the same level as the application. Provide **an explanation of the attack** and the secret **flag**.

3.2. Which CWE(s) did you exploit here?

3.3. How would you fix the problems in the code? There might be more than one thing to fix here.