MINES Saint-Étienne

Une école de l'IMT

**UP: Sécurité des Réseaux Informatiques pour le Web**
**RUP: Raphael VIERA**

# Instructions for the Report

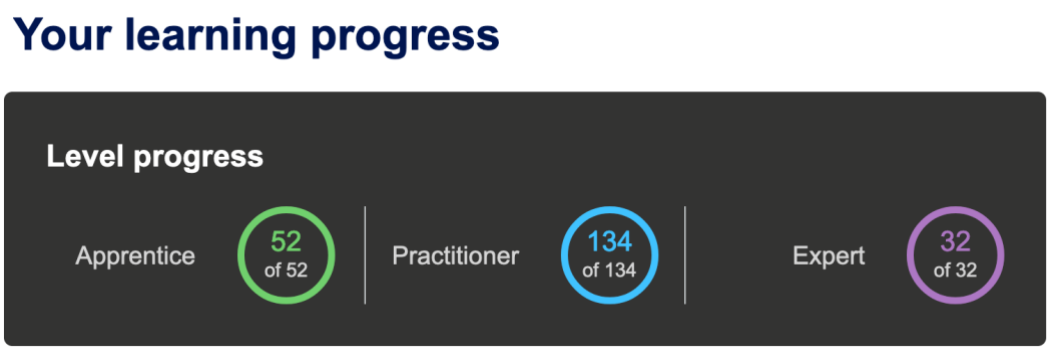
## General instructions

Two persons per report (three if approved by the teacher)
20 pages minimum, 40 maximum
Report written in French or English
Submission on eCampus on .zip format only
Deadline: 15/11/2023 23h59min

## What should the report contain?

1) Your credentials to access your account(s) on PortSwigger. Paste a screenshot of your learning progress (see image below). Bellow an example of 100% completed labs. **You must complete at least 70% of the labs given in class (52 or more).**

**Labs are individual (one login on PortSwigger per person)!**



2) Pentest two vulnerable web applications:
    a. DVWA (Damm Vulnerable Web Application)
        i. Installation instructions: https://github.com/digininja/DVWA
    b. OWASP Mutillidae II:
        i. Installation instructions: https://github.com/webpwnized/mutillidae
3) Develop a vulnerable application
    a. PyFlaSQL
        i. Download : https://gitlab.emse.fr/raphael.viera/pyflasql

## Report scale (Barème)

Total points: 20 points

**PortSwigger labs (8 points):**

At least 52 labs solved: 8 pts
Between 30 and 52 labs solved = 4 pts
Between 5 and 30 labs solved = 1 pt
Bellow 5 labs solved = 0 pts

**Report structure (1 point):**

The report should have a similar structure as shown below:

1) Introduction
2) PortSwigger Credentials
    a. Screenshot of the “Level Progress”
    b. Username and password for each student
3) Vulnerable application: DVWA
    a. Vulnerability 1 (eg. SQL Injection)
        i. Description
        ii. Exploiting
            1. Level low
            2. Level medium
            3. Level High
        iii. Countermeasure
    b. Vulnerability 2
        i. …
    c. Vulnerability 3
        i. …

4) Vulnerable application: Mutillidae II
    a. Vulnerability 1
        i. Description
        ii. Exploiting
            1. Level low
            2. Level medium
            3. Level High

iii. Countermeasure
b. Vulnerability 2
i. …
c. Vulnerability 3
i. …

5) Vulnerable application: PyFlaSQL
a. Vulnerability 1
i. Description
ii. Implementation
iii. Exploiting
1. How to
2. Countermeasure
b. Vulnerability 2
i. …
c. Vulnerability 3
i. …

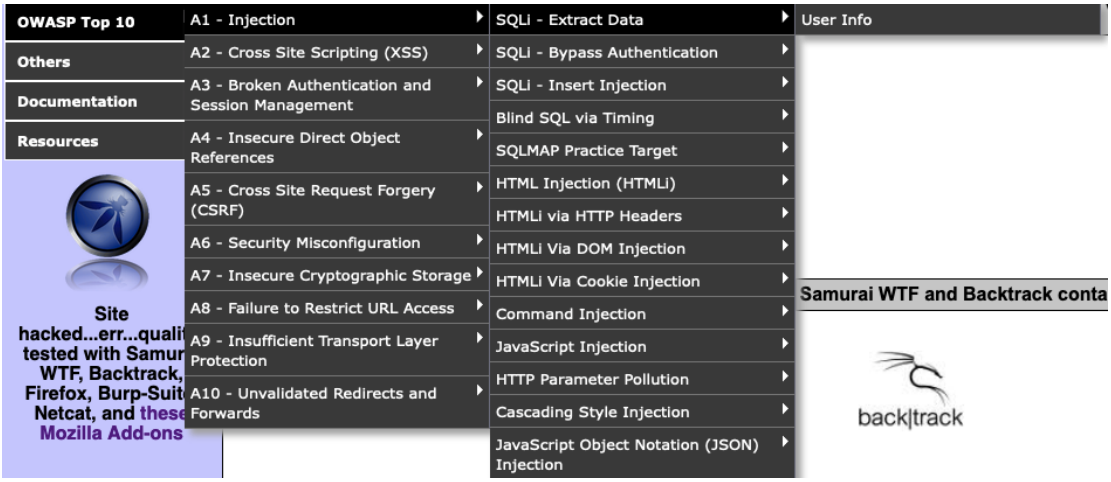
6) Conclusions

**DVWA (3 points):**

This assessment requires the exploitation of at least 3 distinct vulnerability categories, with a reward of 1 point per successfully exploited category. Your objective is to attempt exploitation across all security levels within these three categories, ranging from low security level to high security level. In your report, please include your findings on whether you were able to successfully exploit each category and provide insights on the necessary steps to mitigate and remediate the identified vulnerabilities.


**Mutillidae II (3 points):**

This assessment requires the exploitation of at least 3 distinct vulnerability categories (e.g. A1 – Injection > SQLi – Blind SQL via Timing, then A2, then A3), with a reward of 1 point per successfully exploited category. Your objective is to attempt exploitation across all security levels within these three categories, ranging from security level 0 (Hosed) to security level 5 (secure). In your report, please include your findings on whether you were able to successfully exploit each category and provide insights on the necessary steps to mitigate and remediate the identified vulnerabilities.




While it is permissible for one type of vulnerability to be shared among the two applications, the ideal scenario is to have all vulnerabilities within each application be distinct. In other words, only one common vulnerability type may exist across the two applications, but we aim for the greatest possible diversity in vulnerability types for comprehensive assessment.

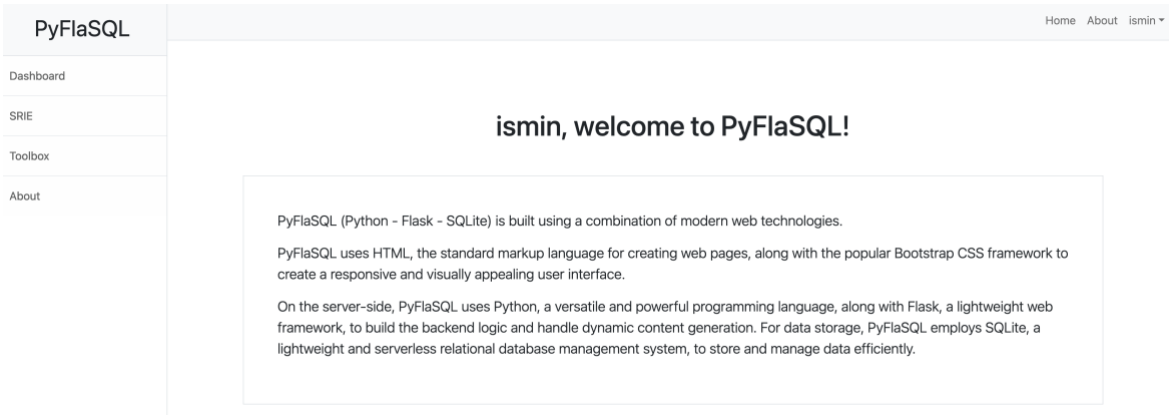
You can find tutorials on the internet on how to attack these applications.

**PyFlaSQL (5 points):**

In this project, your task is to enhance your understanding of software vulnerabilities by exploring and exploiting security loopholes in an application (PyFlaSQL). You have the option to work with the existing application and either add new functionalities or manipulate existing ones, introducing vulnerabilities such as allowing file uploads restricted to .jpg but enabling the upload of potentially harmful formats like .php using Burp Suite or other tools. Additionally, you can experiment with SQL injection or implement various other vulnerabilities of your choice.

Your objective is to implement at least three OWASP top 10 vulnerabilities (https://owasp.org/www-project-top-ten) and provide detailed explanations of the code changes you made. Clearly outline how you explored and exploited the vulnerabilities, including providing screenshots from tools like Burp Suite and web browsers, similar to the approach used in the labs. After successfully exploiting the vulnerabilities, your task is to propose effective countermeasures to mitigate these security risks.

The ultimate goal of this project is to deepen your understanding of how vulnerabilities emerge during the application development process. By comprehending these vulnerabilities and their exploitation methods, you will be better equipped to develop secure systems in the future. This hands-on experience will enhance your ability to write robust and secure code.


PyFlaSQL
Home
About
ismin
Dashboard
SRIE
Toolbox
About
ismin, welcome to PyFlaSQL!
PyFlaSQL (Python - Flask - SQLite) is built using a combination of modern web technologies.
PyFlaSQL uses HTML, the standard markup language for creating web pages, along with the popular Bootstrap CSS framework to create a responsive and visually appealing user interface.
On the server-side, PyFlaSQL uses Python, a versatile and powerful programming language, along with Flask, a lightweight web framework, to build the backend logic and handle dynamic content generation. For data storage, PyFlaSQL employs SQLite, a lightweight and serverless relational database management system, to store and manage data efficiently.