Assignment 4

Web-based Vulnerabilities & Attacks

This assignment contains a mock Web application with a back-end database, consisting of two tables, namely "*users*" and "*items*". The application supports the following functionalities:

- Login as a simple user (username: "user").
- Search for items based on their name and display their information.
- Login as an administrator (username: "superadmin").
- View files.
- Logout.

The application, however, also contains six distinct vulnerabilities:

- Two separate SQL injection vulnerabilities.
- A reflected cross-site scripting (XSS) vulnerability.
- A DOM-XSS vulnerability.
- A local file inclusion (LFI) vulnerability.
- An open redirect vulnerability.

Tasks

1. Bypass the initial login page using an SQL injection payload and login as "user".
2. Identify and exploit the DOM-XSS vulnerability.
3. Identify and exploit the reflected XSS vulnerability.
4. Misuse the item search functionality to retrieve data from the "users" DB table and acquire the admin's password.
5. Login in as the administrator.
6. Identify and exploit the open redirection vulnerability by triggering a redirect to http://example.com/.
7. Identify and exploit the LFI vulnerability to find the secret flag.

How to run locally

1. Download the .zip file, extract its contents and `cd public/`.
2. You will need to have Python3 installed in your system.
3. Run `python3 -m pip install -r requirements.txt`.
4. Run `./run.sh`.
5. The application will be running under `http://127.0.0.1:8080`.

Notes

1. The remote application, which includes the real flag, runs on **http://139.91.71.5:11337** and will remain online for the duration of the assignment.
2. The application has three pages:
    a. User login: “/”
        i. User dashboard and search: “/dashboard”
    b. Admin login and dashboard: “/admin”

    The dashboard is **inaccessible** unless you first login as a user.
3. It is highly recommended to study the provided source code and understand the application’s logic. However, the passwords and the flag are fake. You will need to retrieve them from the real application (see note #1). The credentials (username:password) for the mock application are:
    a. user:user
    b. superadmin:superadmin
4. The XSS payloads can be anything you want, as long as it is executable JavaScript code, e.g., alert(“xyz”).
5. If you want to reset your session and progress, delete the “*session”* cookie and refresh your browser.
6. When you have exploited all vulnerabilities locally, try your payloads in the real server.
7. You only need to provide the password to login; usernames are set automatically.
8. The two database tables and their structure are provided below:

Table: **users**

ID	**username**	**password**
0	superadmin	XXXXXXXXX
1	user	YYYYYYYYY

Table: **items**

ID	**name**	**category**	**price**
0	item0	Music	123
…	…	…	…
99	item99	Music	456

Do’s and Do not’s

1. The only files you need to examine are:
    a. ‘app/app.py’
    b. ‘app/static/js/greet.js’
    c. ‘app/templates/dashboard.html’
2. Do not edit anything in the ‘db’ directory.
3. **Do not use automated tools to exploit the vulnerabilities; you have to do it manually. You are request limited (500/hour).**

Deliverables

You need to submit a **PDF file** that includes:

- A **brief** explanation of your approach to exploit **each** vulnerability.
- The specific **payloads** you used and **where** you injected them. For instance:
    - URL: http://example.com/page.html
    - Form field / GET parameter: input_name
    - Payload: <... attack payload …>
- The **secret flag** you found in the admin’s page.