IAS-Lab

Implementation of a website with *security by design* for the laboratory project of Application Security (IAS) at Universitat Politècnica de Catalunya (UPC).

Table of contents

Framework

The project is based on **Django** on top of a **dockered** database using **Mongo**.

Django

We decided to use Django (a python framework) for web development. We will implement our project using Model–view–controller (MVC) architecture patterns.

The main reason for this decision is that we have some experience with it. However, another reason to use Django is because of some security features being implemented by design, such as the hashing of passwords or session management.

Docker

We defined a Docker container to build the development environment and avoid the cross operating system environment errors by abstracting the deployment environment.

MongoDB

About the database, we decided to use MongoDB, a distributed and document oriented database. This is due to it being currently used all over the world, meaning it has present relevance in the community. By using this database we also want to get some experience and get a hold of the MongoDB way.

Application

The application will be a web to upload PDF files (books).

Login

The page where you will be redirected once you access the web will be the login.

There you will be able to access with user and password or login with Google (if you had previously registered an account with the email associated to the Google account).

You can fail the login 5 times, once you fail them you will have to wait for 15 minutes as your login attempt will be blocked.

There will be also a button to register a new user in the login webpage.

Register

On the register page there will be a classic form:

- Username
- Email (x2)
- Password (x2)
- Invitation code
- CAPTCHA

The invitation code will be a code associated with an existing user in order to avoid unknown users to register and be only able to register if you know someone who is already a member of the webpage.

The CAPTCHA will prevent bots from forcing invitation codes.

The provided passwords will be hashed when they are stored in the database (done automatically by the framework).

Home

After a successful login the front page will be displayed. If this URL is tried to be accessed without login you will be redirected automatically.

The session will be managed using Django session manager.

The frontage will present a table with information of the books, such as title, ISBN, publication date…

There will be also a button to download the books, a button to upload new books but only available for people with administrative rights and a button to the user profile where the user can change its password In order to change the password the old password must be provided.

If a POST is tried to this link with a user who is not admin the POST won't be processed.

Upload

The upload section will be a form where the information of the book will be provided, only being mandatory the title of the book.

There will be also a field to select the file that you wish to upload. This file will be verified in the backend checking the magic number and the file extension. If the magic number or the extension is not pdf, epub or mobi the POST will be rejected.

The file size will be also checked not allowing files with a size higher than 100 MB.

User profile

Your name will be shown and there will be a list of your permissions.

Moreover, there will be also 1 or 2 fields containing the invitation codes. If you are a normal user only one invitation code will be displayed in the form of UUID or similar. If the user is an admin 2 codes will be displayed the normal code and the admin code. Depending on which one is used in registration time will determine if a user is a normal user or admin. With this method only an admin can invite another person with admin permissions.

The invitation code displayed in the profile will be different for each user.

This will allow the creation of a graph to track the invitation tree.

Other details

All the fields (in Registration, Upload and Login) will be serialized to avoid code injections. The serialization will happen at database driver time so any input will be possible but just before processing they are serialized automatically.

The web server and database will be dockered in order to deploy it more easily. A VPS will be used in the deployment. The machine is hosted by Digital Ocean.

Proposed Modifications

Modifications proposed in last laboratory class that will be added to the final version of the project (along with the original missing ones).

DDoS Protection

We will use the DDoS Cloudfare mitigation to be protected against this kind of attacks. This will allow us to be always up and running and offering a reliable service.

Malicious users graph

All new users register in the system with an invitation code that belongs to an existing user. Then, we can model the relation between users with a directed graph showing who invited who.

This allows us to have a clear view of the status of our platform and to apply extra security measures. For example, we can detect that a user is inviting users that behave incorrectly or that always end up banned. After detecting this, we can revoke the invitation key of that user.

Install

To start running the server we reccomend the following steps:

1. Create a **virtual environment**

```
$ sudo apt install virtualenv
$ virtualenv --python=python3 venv
$ source venv/bin/activate
```

2. Clone the **Github repo**

```
$ git clone https://github.com/muniategui/IAS-Lab.git
$ cd IAS-Lab/
```

3. Install **requirements**

```
$ pip install -r requirements.txt
```

4. Get the **Django database** ready

```
$ python manage.py makemigrations
$ python manage.py migrate
```

5. Create a **superuser** to administrate the site

```
$ python manage.py createsuperuser
```

6. **Run** server

```
$ python manage.py runserver
```

Interesting URLs

Site	URL
Web	http://127.0.0.1:8000/
Admin	http://127.0.0.1:8000/admin/

Basic features checklist

- [x] User registration
- [x] User login
- [ ] Register contents in the application (PDF)
- [ ] Access control to contents
- [ ] Manage access permissions to contents

Security features checklist

- [x] Cross Site Scripting (XSS) protection
- [x] DB injection protection
- [x] Hashed passwords
- [ ] Create and install a server certificate
- [x] Cross site request forgery (CSRF) protection
- [ ] Content encryption
- [x] CAPTCHA
- [x] Bruteforce prevention at login

For noob developers (we have some)

Get ready your user settings:

```
$ git config --global user.name "USERNAME"
$ git config --global user.email "MAIL"
```

To make changes to one file do:

```
$ git pull
$ git commit FILENAME -m "MESSAGE"
$ git push
```

Multifile:

```
$ git pull
$ git commit * -m "MESSAGE"
```

```
$ git push
```

Errors? Use:

```
$ git reset --hard
```