Final Project - Secure File-Sharing Server

Scenario

Your organisation intends to build a file-sharing service to be used by their collaborators. They have assigned to your group, the secure software development team of the organisation, the role of designing and implementing a secure solution.

The service shall be developed in the form of a modern web page, with:

- **client**: a web page frontend that supports many users and allows users to upload their files, share them with each other and perform file maintenance operations.
- **server**: a backend with REST endpoints that can be called by the client application. Advanced users may directly call the REST endpoints. The backend shall include:
  - a global SQL (e.g., mySQL), or NoSQL (e.g., REDIS) database storing relevant user and file information. You can make use of doctor to install such databases.
  - a local directory structure per user, in which user files are stored in (a projection of) the server's filesystem.

Requirements

You shall design and build your service according to the following general requirements:

- A. **Interface**:
  - Each user shall have a local directory structure, much like the Unix file system. Users shall be able to:
    - Perform simple operations such as creating, deleting or renaming files via the graphical web frontend;
    - Perform batch or more advanced operations via a Unix-like shell environment over their local directory structure;
    - Both modes of operation may be supported by a backend RESTful API reminiscent of, e.g., (a simplification of) the Google Drive API.
  - Users shall be able to share files/directories among each other.
  - Users shall be able to publish a personal web page reminiscent of, e.g., (a simplification of) GitHub pages. You are free to choose a design. For instance, a user may have a separate public HTML folder or all his/her directory structure may be rendered via the backend's HTTP server.
- B. **Authentication**:
  - Users have to login in the web frontend before accessing their directory structure.
  - Operations on a user's directory structure require authenticated users.
  - You shall have a minimal authentication method, e.g., password-based. If you see fit you may additionally use secure HTTPS communication for authenticated users. Note, however, that authentication and secure communication are **not** the focus of this project.
- C. **Access control**:
  - A user shall only have access to files/directories that he/she owns or that have been shared with him/her.
  - A user may share a file/directory with other users with view or edit permissions.
  - A user's personal web page is public.

Design

Your team shall design the software components needed for the service, coupled with the security mechanisms to satisfy the requirements outlined above. You shall seek to adopt a security by design software development methodology. A natural direction is to design a sytem architecture, specify actors and use cases, and identify security threats and assumptions before writing a single line of code.

The security goal of this service is to protect against external client-side attackers or malicious users. You may assume that, aside from vulnerabilities in your design or implementation, the server is trusted.

Implementation

You shall implement a functional service that meets the above general requirements and instantiates your proposed secure design. You can use any technology and programming language for the server/client code that you see fit (e.g.,

C/C++, Java, JavaScript, Python, Ruby, ...).

Beyond advanced functionality, familiarity or ease of deployment, your choice shall take into consideration the reliability and the security guarantees offered by each technology. Secure and reliable software procurement is also a job of security developers and teams.

Analysis

You shall also seek to demonstrate that your implementation respects your design and fulfils its security requirements. You shall describe the security analysis methodology that you have employed to validate your implementation and corroborate the overall security of your system. This includes, e.g.:

- security guarantees offered by the programming languages;
- adopted secure software design patterns;
- security analysis of dependencies or external libraries;
- adopted mitigations for common vulnerabilities;
- security testing methodologies that you have put in place;
- source code analysis tools that you have considered adequate.

Report

The project report must describe your design, implementation and analysis. The report shall primarily focus on design and analysis decisions, in detriment of implementation details.

Remember that software design, implementation and analysis are not independent phases but are part of a continuous software development process. Any lessons that you have learned along the way, such as implementation details that have led you to revisit your initial design or design/implementation vulnerabilities that had not initially predicted but have found through later analysis, are valuable information to include in the project report.

Presentation

At the end of the semester, each group will present their assignments during classes. The presentation shall highlight the most relevant security-oriented details of the design/implementation/analysis, as in the written report, and showcase common uses of your service.

Grading

Each component will have the following percentage in the project's final grade:

Component	%
Design	25
Implementation	25
Analysis	30
Presentation & Report	20