Homework 2: Secure Full-Stack Web Apps (React + FastAPI + Docker)

Due: Thursday, October 23rd, 2025

Estimated completion time: 1–2 hours
Stack: Python (FastAPI), React, Axios, Redis, Docker

Multiple Choice (10 pts each)

1. Which header allows a frontend on a different domain to make requests to a backend?
   2 A) Authorization
   1 B) Access-Control-Allow-Origin
   4 C) X-CSRF-Token
   3 D) Content-Type

2. What is the primary purpose of a CSRF token?
   4 A) Encrypt user passwords
   1 B) Prevent cross-site request forgery
   2 C) Store session info in the browser
   3 D) Track analytics

3. In React, which Axios option ensures cookies are included in cross-origin requests?
   4 A) withCookies: true
   2 B) credentials: include
   3 C) crossDomain: true
   1 D) withCredentials: true

4. What is a major advantage of storing sessions in Redis?
   1 A) Allows sessions to persist across multiple servers
   3 B) Automatically encrypts passwords
   4 C) Replaces the need for CSRF tokens
   2 D) Prevents CORS errors

5. Which React hook is most appropriate for fetching user info after login?
   1 A) useEffect
   3 B) useMemo
   4 C) useRef
   2 D) useState

6. Which Docker command builds an image from a Dockerfile?
   3 A) docker run
   1 B) docker build
   4 C) docker start
   2 D) docker compose

7. What happens if a CSRF token is reused in a one-time token system?
   3 A) It works as usual
   1 B) The request is rejected
   4 C) The session is deleted automatically
   2 D) The token is refreshed

8. In React Router v6, what does <Navigate to="/home" replace /> do?
   2 A) Renders a new component in place
   1 B) Redirects the URL to /home without adding a history entry
   3 C) Replaces the component props
   4 D) Updates the Redux store

9. Which statement is TRUE about CORS?
   2 A) Cookies can be sent with Access-Control-Allow-Origin: *
   1 B) The server must explicitly allow origins to accept credentials
   3 C) CORS is only needed for POST requests
   4 D) CSRF tokens replace the need for CORS

10. Why is setex useful when storing CSRF tokens in Redis?
   3 A) It encrypts the token
   1 B) It sets a time-to-live (TTL) for automatic
   expiration
   2 C) It hashes the session ID
   4 D) It enables cross-origin requests

True/False (5 pts each)

1. CSRF tokens should be unique for every state-changing request. True
2. Using Access-Control-Allow-Origin: * is safe if you also send cookies in CORS requests. False
3. React state is persistent across page reloads without using localStorage or cookies. False
4. Asession stored in Redis can expire automatically using a TTL (time-to-live). True

5. In FastAPI, the Depends() function can be used to inject session management into multiple endpoints. True

---

Short Answer (6 pts each)

1. Explain the difference between session-based authentication and token-based authentication.

Session-based authentication creates and stores a session and gives the client a cookie containing a session ID while Token-based authentication server issues a signed token to the client

2. Why is it important to include CSRF tokens in headers when making POST requests with Axios?

It ensures that the request from the site is not malicious . It protects users from the attackers

3. What is the role of withCredentials: true in Axios requests?

It makes Axios to include cookies that is required for the session-based authentication site

4. How does using Redis for session storage improve security and scalability?

It stores session in the memory and to allow multiple servers to share the same sessions.

5. In Docker, why might you use a docker-compose.yml instead of running each container individually?

It simplifies configuration and manage dependencies for FastAPI, React, etc.

6. What is the purpose of a redirect route like <Route path="/" element={<Navigate to="/home" replace />} /> in React Router?

It redirect users from the root path to the /home route when the user visit the base URL.

Code Reading (5 pts each)

Given the FastAPI route below:

```
@app.post("/update-profile")
async def update_profile(session: Session = Depends(get_session),
data: dict = Body(...)):
    token = data.get("csrf_token")
    if not await CSRF.verify_csrf(session, redis_store, token):
        raise HTTPException(status_code=403, detail="Invalid CSRF
token")
    session._data["profile"] = data.get("profile")
    await session_manager.save_session(session)
    return {"success": True}
```

1. What happens if the client submits an invalid CSRF token?

   it will display a 403 error and the request will be rejected

2. How does the Depends(get_session) function help this endpoint?

   It provides access to the user’s session within the route

3. If the session is not saved after updating _data["profile"], what potential problem occurs?

   It will not update the profile and the next request will load the old data

Code Implementation (10 pts each)

1. Fillin themissingcodetomakethefollowingAxios request include cookies and CSRF token:

```
importaxiosfrom"axios";

constupdateProfile=async (profileData, csrfToken) =>{
    constresponse=await axios.post("/update-profile",
profileData, {
       //TODO:include cookies and CSRF token
       csrf_token: csrfToken,
       },
       {
       withCredentials: true,
       headers: {
         “Content-Type”: “application/json”,
       “X-CSRF-Token”: csrfToken,
       },
       }
       }:
});
returnresponse.data;
};
```

2. Write a React snippet for a login form that:

● Uses useState for email and password.

● Sends a POST request to /login using Axios.

● Handles errors in state and displays them.

```
import React, { useState } from "react";
import axios from "axios";

const LoginForm = () => {
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [error, setError] = useState("");

const handleSubmit = async (e) => {
e.preventDefault();
setError("");

try {
const response = await axios.post(
"/login",
{ email, password },
{ withCredentials: true }
);
console.log("Login successful:", response.data);
} catch (err) {
setError(err.response?.data?.detail || "Login failed");
}
};
```