Table of Contents

Code reviewed

Team	Score	Observations
3	1	The code was very small and simple. There was no effort in disguising any error/vulnerability.

Code developed

Test team evaluation

Team	Score	Observations
1		

Vulnerabilities

- **SQL Injection** - During registration, a query for the existing usernames is made to prevent username collision. This SQL query however, does not sanitize the user input, allowing the system to expose all usernames in said header field.
- **JWT Token Forgery** - Ideally, servers should only use a limited whitelist of public keys to verify JWT signatures. However, this system trusts any key that's embedded in the `jwk` parameter. This allows any attacker to self-sign tokens, allowing for token forgery.
- **OS Command Injection** - If a user with the admin role accesses the `/studentsByClass` endpoint, the system will perform a backup of the database. The content of the token's `role` field is concatenated, without sanitization, to the backup file via OS command, allowing for OS command injection via token forgery.

From OWASP's Secure Coding Practices Checklist the three vulnerabilities implemented fall under the following categories:

- Validate all client provided data before processing;
- Contextually sanitize all output of un-trusted data to queries for SQL, XML, and LDAP;
- Sanitize all output of untrusted data to operating system commands;
- Use only trusted system objects, e.g. server side session objects, for making access authorization decisions;
- Protect secrets from unauthorized access.