Risk Analysis

Name: Tony Jiang

Semester: 6

Introduction

This document outlines the security risks that can occur in the music trivia web-based game and evaluates whether they are covered in the project and how they would be addressed. Using the top 10 OWASP also help define some security risks for this project.

Risk identification

The "impact" is how long it would take to implement it in the project. Where 1 is low and 5 is high.

The "likelihood" is how likely the security risk can happen Where 1 is low and 5 is high.

Risk ID	Risk Description	Impact(1-5)	Likelihood(1-5)	Implemented
1	Unauthorized access to the song management section of the website.	3	2	Yes
2	Storage of user passwords in the database without encryption.	1	1	Yes
3				
4				
5				
6				
7				
8				
9				
10				

Mitigation plan

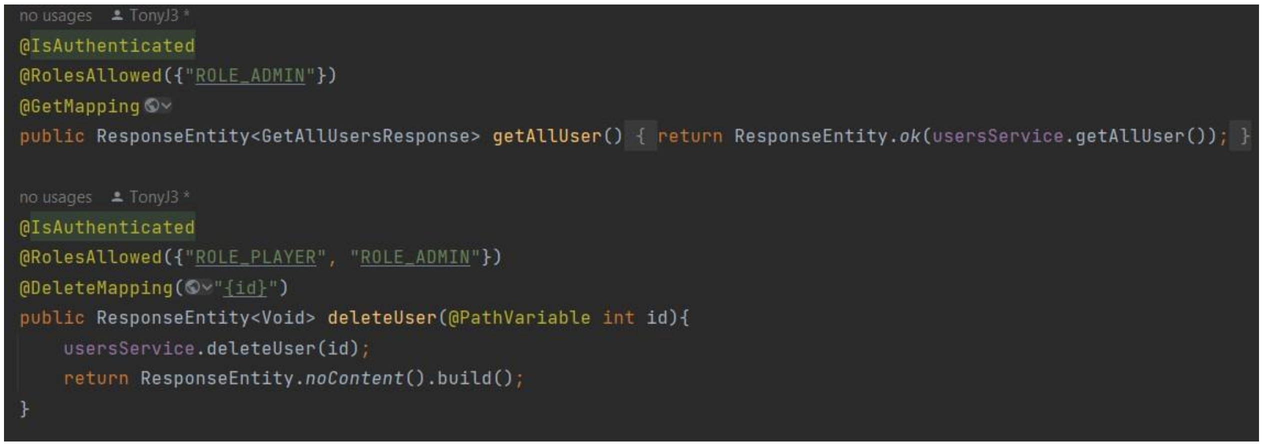
This section outlines how the identified security problems will be addressed. It includes implementation strategies or solutions to solve the problem. These measures are to be taken into consideration.

1. Unauthorized access to the song management section of the website.

Mitigation: Implement user roles.

Evidence:

Each function in the controller is associated with a specific user role that determines who can access that function. If a function is not assigned a role, it can be accessed by anyone.



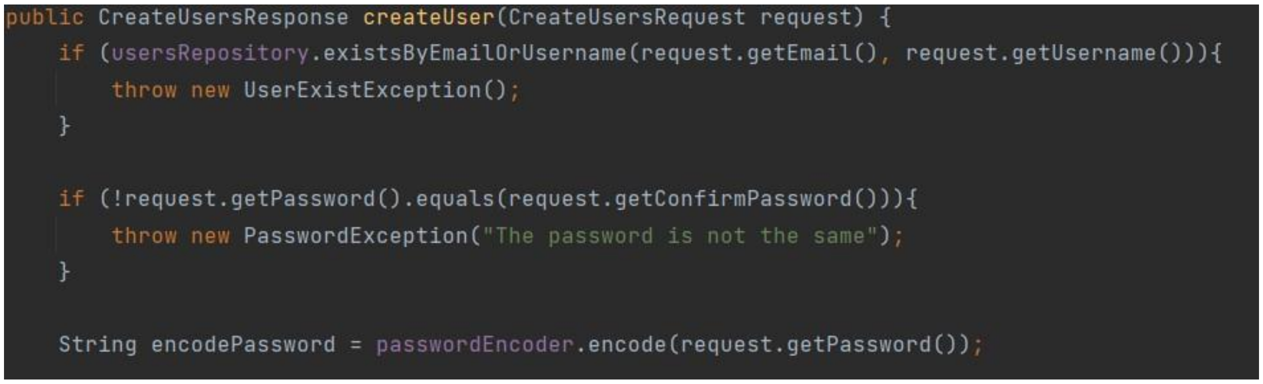
```
no usages   TonyJ3 *
@IsAuthenticated
@RolesAllowed({"ROLE_ADMIN"})
@GetMapping
public ResponseEntity<GetAllUsersResponse> getAllUser() { return ResponseEntity.ok(usersService.getAllUser()); }

no usages   TonyJ3 *
@IsAuthenticated
@RolesAllowed({"ROLE_PLAYER", "ROLE_ADMIN"})
@DeleteMapping("{id}")
public ResponseEntity<Void> deleteUser(@PathVariable int id){
    usersService.deleteUser(id);
    return ResponseEntity.noContent().build();
}
```

2. Storage of user passwords in the database without encryption.

Mitigation: Implement password hashing.

Evidence:

I have implemented the password hashing when you sign up in the application. I use the bcrypt algorithm hashing. Bcrypt is wildly use that doesn't use a lot of memory and performance to hash and it's one of the moderate hashing security.



```
public CreateUsersResponse createUser(CreateUsersRequest request) {
    if (usersRepository.existsByEmailOrUsername(request.getEmail(), request.getUsername())){
        throw new UserExistException();
    }

    if (!request.getPassword().equals(request.getConfirmPassword())){
        throw new PasswordException("The password is not the same");
    }

    String encodePassword = passwordEncoder.encode(request.getPassword());
```