Issue Description:

The current implementation allows both confidential and public clients to perform token refresh grants. However, for security reasons, it's important to restrict this functionality to confidential clients only. This proposal outlines the necessary changes to achieve this and enhance the overall security of the system.

Proposed Solution:

The proposed solution involves implementing a mechanism that verifies the client type before allowing the refresh token grant. This can be achieved by leveraging the OAuth 2.0 `client_id` and `client_secret` combination to differentiate between confidential and public clients.

Implementation Steps:

- **Identify Client Type:**

  Modify the `refreshTokenRequestValidator` to validate the client type making the refresh token request. This can be accomplished by querying the `applicationRepository` for the client details based on the provided `client_id`. If the client is not confidential, an exception will be thrown, indicating that only confidential clients are allowed to perform token refresh.

- **Validation Logic:**

  Update the `refreshTokenRequestValidator` to include the validation logic for client type. Here's a simplified example of how this could be implemented:

```
public void validate(String grantType, String clientId, String
refreshToken) {
    // Perform general validation checks (grantType, refreshToken,
etc.)

    final var client =
        applicationRepository.findByClientId(clientId)
            .orElseThrow(() -> new BadRequestException("application
not found for clientId: " + clientId));

    // Check if the client is confidential
    if (!client.getClientType().equals(ClientType.CONFIDENTIAL)) {
        throw new UnauthorizedException("Only confidential clients
are allowed to refresh tokens.");
    }
}
```

- **Client Entity:**

  Ensure that the `Application` entity includes a field representing the client type (e.g., `clientType`). This field could be an enum with values like `CONFIDENTIAL` and `PUBLIC`.

- **Unauthorized Exception:**

  Define a custom exception class, such as `UnauthorizedException`, to handle cases where a public client attempts to perform a token refresh. This exception class can be created by extending `RuntimeException` or a more specific exception class.

```
public class UnauthorizedException extends RuntimeException {
    public UnauthorizedException(String message) {
        super(message);
    }
}
```

Benefits:

Implementing this solution will enhance the security of the token refresh process by ensuring that only trusted and confidential clients can perform token refreshes. Public clients, which may not securely store client secrets, will be denied access to this functionality.

I hope that this solution will address the issue effectively and contribute to the overall security posture of the system. If approved, I would be glad to work alongside a seasoned contributor and take on the task of implementing this solution and creating a pull request for the proposed changes. I am new to open source contribution space, hence would need a bit of hand holding initially.