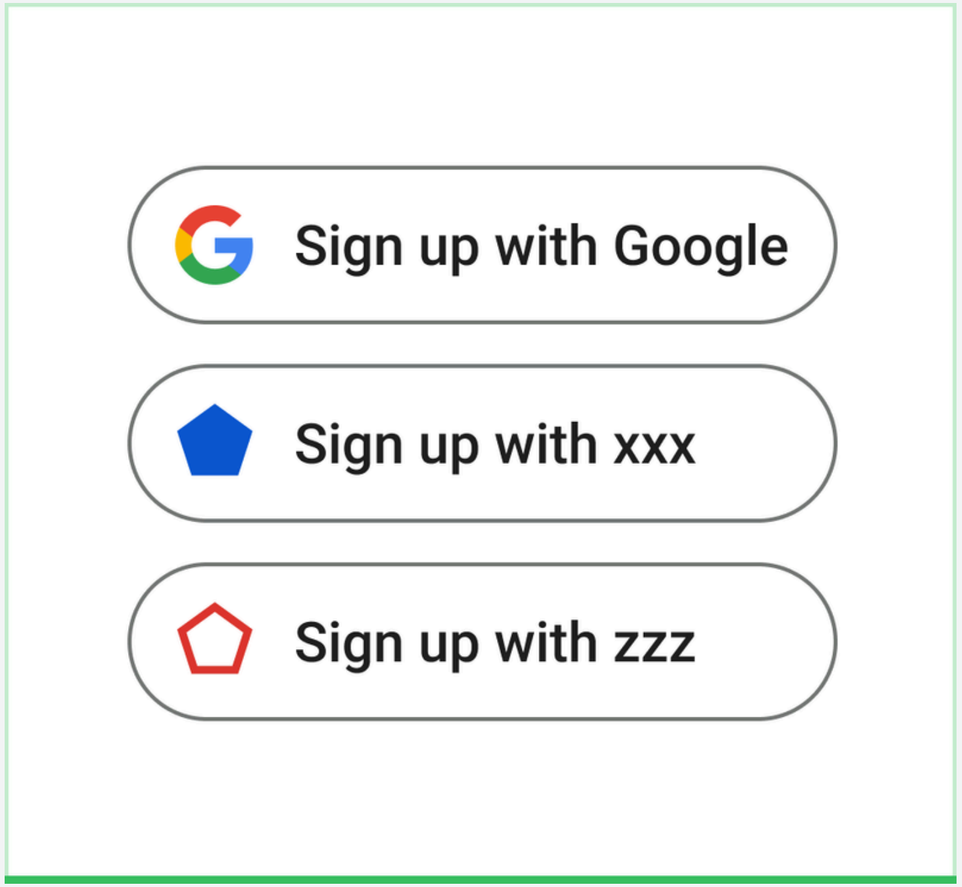

OAUTH-BASED
AUTHENTICATION IN WEB APPLICATIONS

INTRODUCTION

- OAuth (Open Authorization) is a secure token-based authorization framework.
- Allows third-party apps to access user information without sharing passwords.
- Commonly used by platforms like Google, Facebook, GitHub.



KEY ROLES

01

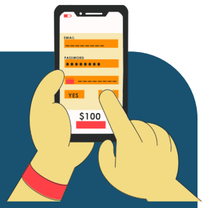
Resource Owner

The user who owns the data and is granting access (typically the person logging in)

02

Client

The app requesting access to the resource on behalf of the user



03

Authorization Server

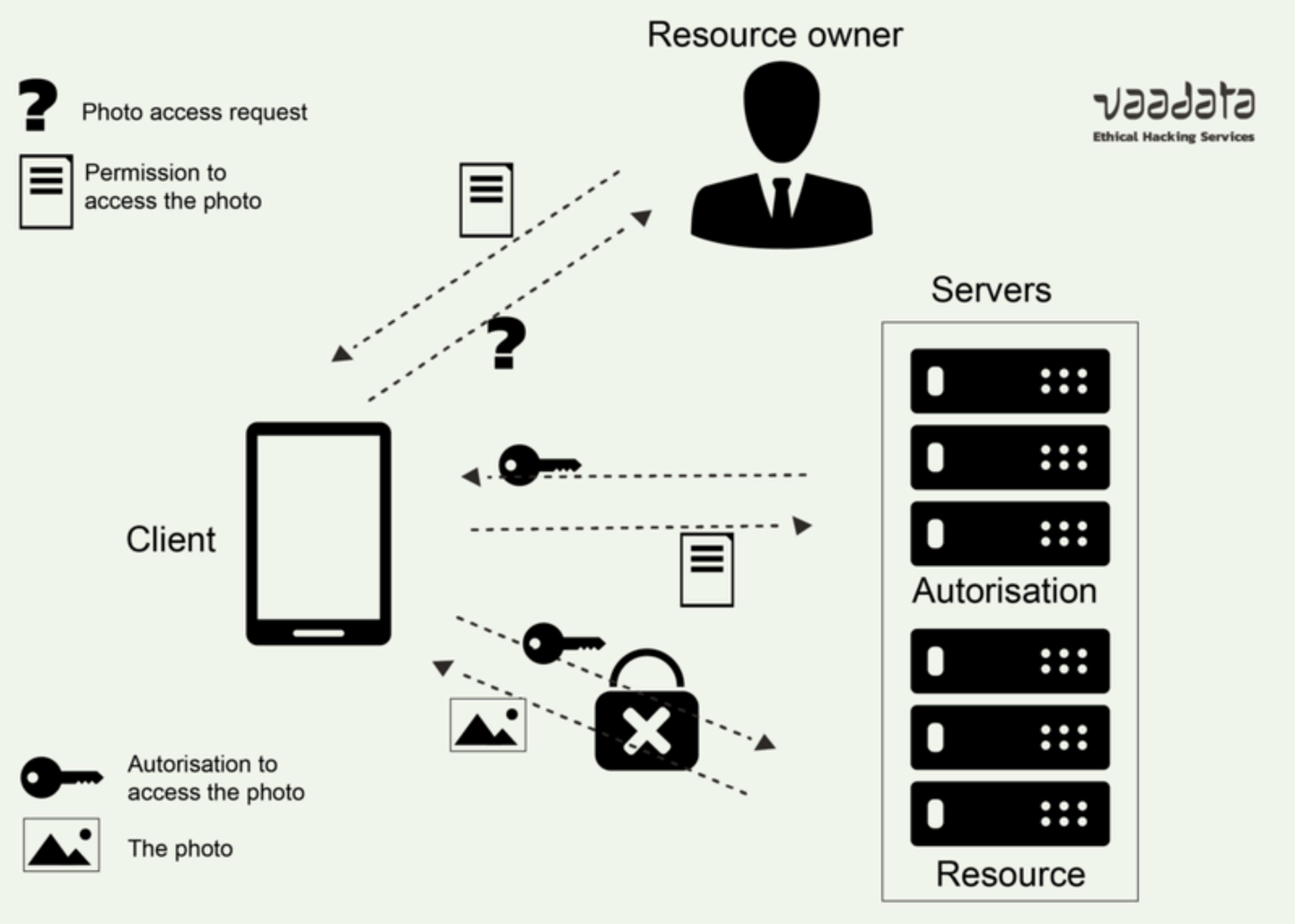
The server responsible for authenticating the user and issuing tokens

04

Resource Server

The server that holds and serves protected user data (requires access token)

WORKING OF OAUTH



IMPLEMENTATION OVERVIEW

Frontend: HTML/CSS/JS for the student portal UI

Backend: Node.js + Express for handling OAuth logic

Endpoints Implemented:

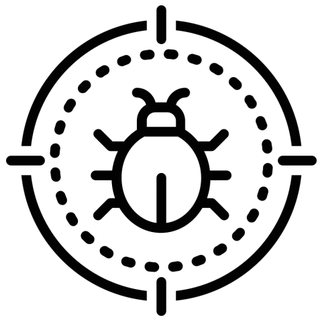
/authorize

/token

/userinfo

Attack Simulation - Token Reuse

- Logged in as Admin, obtained Admin access token
- Logged in again as Student, obtained Student token
- Replaced Student’s token with Admin’s token using Burp Suite
- Gained access to Student resources with Admin token



VULNERABILITIES IDENTIFIED

- No role-based validation of tokens
- Tokens not scoped or bound to users
- Token reuse across roles possible
- No expiration or revocation implemented

MITIGATION STRATEGIES

- Enforce role-based validation at the resource server
- Use OAuth scopes effectively
- Set token expiration and revocation mechanisms
- Bind tokens to user sessions

CONCLUSION

- **Token reuse can lead to unauthorized access and privilege escalation.**
- **Lack of role validation and broad token scopes are major risks.**
- **Secure OAuth requires role checks, token expiration, and scope restrictions.**