LAB 08: Access control vulnerabilities - OAuth2.0

Part 1: Answer the following questions

1. What is OAuth, and how does it facilitate authorization in modern web applications?
    - OAuth, short for "Open Authorization," is an open standard that lets you grant one application (like a website or mobile app) access to your data stored in another application (like your social media account), all without having to share your password with the first application.
    - You want to use an app (called the "client") that needs access to your data on another app (called the "service provider"). For example, you want to log in to a fitness app using your Google account. The client app redirects you to the service provider's login page. You log in using your credentials with the service provider (e.g., Google). Instead of sharing your password, the service provider shows you a list of permissions the client app wants to access (e.g., your name, email, fitness data). You review and grant access if you're comfortable.
    - The service provider sends the client app a temporary "access token" instead of your password. This token lets the client app access your data on the service provider's platform, but only for the specific permissions you granted. The client app uses the access token to securely access your data on the service provider. No password is ever shared with the client app, making it much less vulnerable to security breaches.
2. Describe how consent phishing can be executed in an OAuth context. How can attackers leverage OAuth permissions to gain unauthorized access to user data? Discuss strategies that can be employed to detect and prevent consent phishing in OAuth implementations.

Consent Phishing in OAuth:

- Malicious App Impersonation: Attackers create apps that mimic legitimate services, tricking users into granting permissions to access sensitive data.
- Spoofed Consent Screens: Attackers manipulate consent screens to misrepresent the permissions being requested, leading users to grant excessive access unknowingly.
- Social Engineering: Attackers leverage tactics like urgency or fake rewards to pressure users into hasty authorization decisions.

How Attackers Leverage OAuth Permissions:

- Excessive Data Collection: Attackers harvest sensitive information beyond what's necessary for the app's functionality.
- Data Misuse: Stolen data can be sold, used for identity theft, targeted advertising, or other malicious purposes.
- Account Takeover: Permissions like "email" or "profile" can be abused to compromise user accounts on other platforms.

Strategies to Detect and Prevent Consent Phishing:

- User education ○ Robust App Review Processes ○ Security Features ○ Regular Audits and Monitoring