Why You Should Not Use ROPC for Mobile Applications

Resource Owner Password Credentials flow with public clients is typically used to enable applications to continue to provide login screens. However, there are major security issues.

The Good

- You get tokenized API access
- Easy to migrate legacy applications that relied on Basic authentication

The Bad

- User credentials are exposed to the client application
- The user and client application are indistinguishable from one another
- Client applications can request any scope without the user's knowledge
- You are training your users to be phish able
- Don't expect it to be speedy (you have to perform password hashing after all)
- Flies in the face of the OAuth best practice guidelines (RFC 8252)
- No federation
- No multi-factor authentication
- No single sign-on

The Ugly

Significantly increases the attack surface on user credentials (if the client is compromised, so is the user's entire account)

- Public client applications cannot keep a secret and therefore cannot authenticate themselves
- Token endpoint, therefore, becomes an open endpoint that attackers can use to brute force credentials and therefore needs to handle risks such as account enumeration, and credential stuffing
- The application could be trivially mimicked, and both the user and authorization server would have no way of knowing, allowing an attacker to harvest both user credentials and access tokens.

The Solution

The only secure solution is to follow the best practices detailed by the OAuth working group in RFC 8252. This involves the use of the authorization code flow, PKCE, in-app browser tabs, and custom URI schemes.