Best Practices for Native Applications

Scott Brady

IDENTITY & ACCESS CONTROL LEAD

@scottbrady91   www.scottbrady91.com

Overview

Understand the unique security issues for native applications

See how PKCE can secure public clients

Lean the best current practices for native applications using OAuth

Native Applications

Mobile applications

Desktop application

Native apps = Public clients

Why Can’t We Just Use the Implicit Flow?

Reliance upon redirect URI

No webserver

We aren’t the only ones listening...

Refresh tokens

Codes or tokens - both at risk

The implicit flow should not be used for native applications

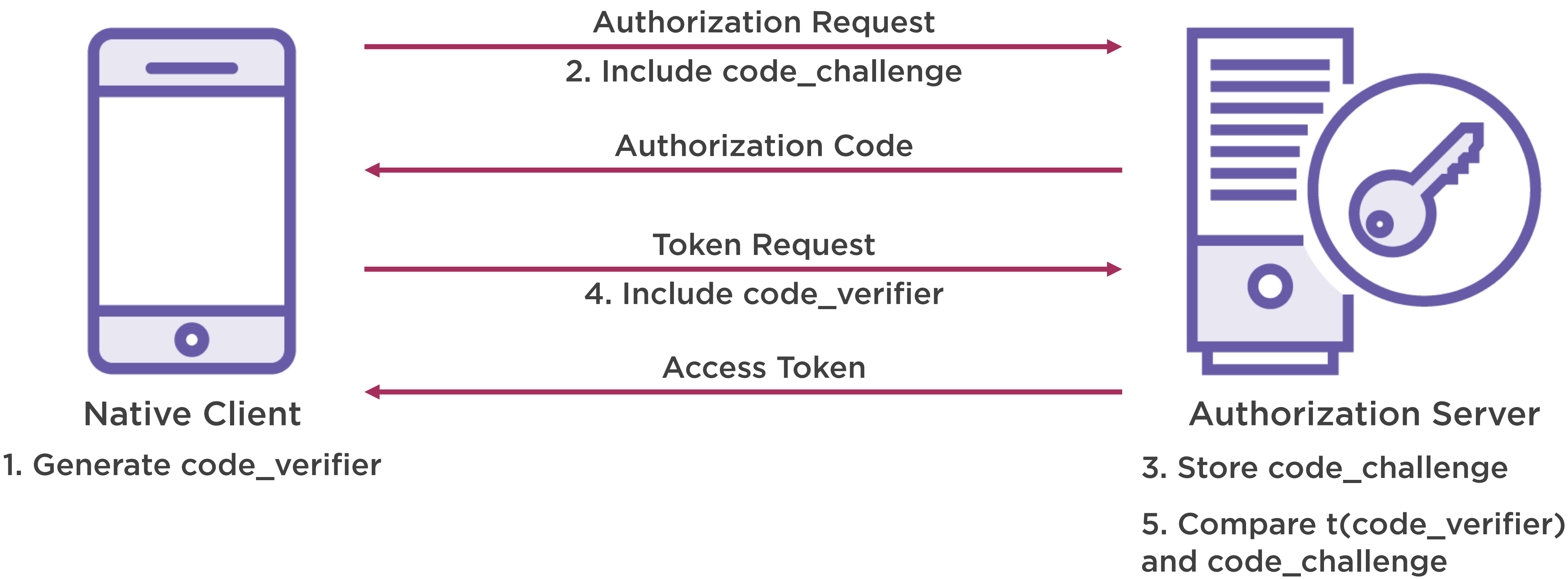
Native Applications

Can	Can’t
Make secure backchannel requests	Receive tokens via the browser
	Keep a secret (or at least won’t be trusted)

Proof Key for Code Exchange (PKCE)

Links the authorization request to the token request

PKCE in Action



Authorization Request

2. Include code_challenge

Authorization Code

Token Request

4. Include code_verifier

Access Token

Native Client

1. Generate code_verifier

Authorization Server

3. Store code_challenge

5. Compare t(code_verifier)
and code_challenge

RFC 7636

Private-Use URI Scheme

e.g. com.pluralsight.ios:/cb

Not available on all platforms

Collisions possible

Redirect URI Options

com.pluralsight.ios:/cb

Private-use URI scheme

HTTPS

Claimed https scheme

Loopback device

Embedded User-Agent (Browser)

Keystrokes visible

Cookies exposed

Authorization process exploitable

Increased attack surface

Encourages phishing

In-App Browser Tabs

Uses the system browser

Password manager integration

Single Sign On

RFC 8525

Demo

OAuth for native apps best practices in action

Summary

Understand the unique security issues for native applications

See how PKCE can secure public clients

Lean the best current practices for native applications using OAuth

Further reading:

- RFC 8252 (OAuth for native apps)