OAU201 - OAUTH SECURITY FUNDAMENTALS

Course Learning Objectives

OAuth Security Fundamentals begins with discussing the problems that OAuth solves and takes a high-level look at how OAuth delegation works. From there, you'll learn how to implement an OAuth flow for a server-side web application, how to implement OAuth in a client-side browser application, and how to implement OAuth flows in native applications. And finally, you'll learn how OpenID Connect extends OAuth with user identity features.

Description

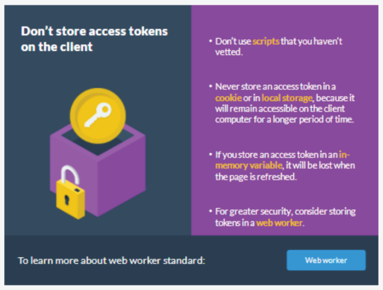
OAuth Security Fundamentals spans five modules. This course is designed for Security Architects and Software Developers. It is recommended that all learners are familiar with the security fundamentals of authentication and authorization, as described in the OWASP Top 10.

Audience

Security Architects
Software Developers

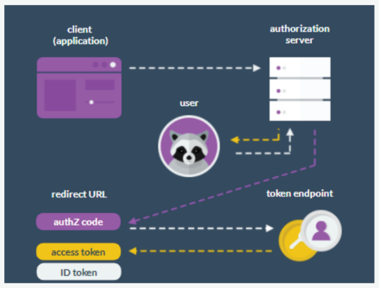
Time Required

Tailored learning - 90 minutes total (approx.)


Don't store access tokens on the client

- Don't use scripts that you haven't vetted.
- Never store an access token in a cookie or in local storage, because it will remain accessible on the client computer for a longer period of time.
- If you store an access token in an in-memory variable, it will be lost when the page is refreshed.
- For greater security, consider storing tokens in a web worker.

To learn more about web worker standard:



OAU201 - OAUTH SECURITY FUNDAMENTALS

Course Outline

1. Laying the Groundwork

- About
- Delegated access with OAuth
- Advantages of OAuth
- Evaluate if OAuth is suitable for your needs
- Identify the type of client for implementing OAuth
- Examples of different clients
- Identifying client
- Choose the correct OAuth flow
- Building a public client
- Building a private client
- Consider OpenID Connect (OIDC) for authentication

* Optional interactivity

2. Securing Server-Based Web Apps

- Registering a client application
- The OAuth registration process
- A registration example with Google APIs
- The client ID and client secret
- Using the authorization code flow
- Choosing an OAuth library
- Example: The authorization request
- Example: The authorization process
- Example: The redirect
- Example: The token request
- Example: Authorized access
- Handle tokens securely
- Hardening the OAuth flow
- Use the Client Credentials Flow if the user isn't present

3. Securing Client-Side Web Apps

- Avoid the implicit flow
- Using the authorization code flow with PKCE
- How PKCE works
- Using PKCE
- Using the state parameter
- Don't store access tokens on the client
- Limitations
- Consider a "backend for frontend" approach

4. Securing Native Apps

- Native applications and OAuth
- Use the Authorization Code Flow for native apps
- Example: An authorization request with AppAuth
- Use a custom URI scheme for redirects
- Claimed HTTPS URIs
- Do not embed a browser view
- Use the system browser
- Use the Device Flow for devices with limitations
- The Device Flow

5. Adding OpenID Connect

- What OAuth wasn't created for
- OIDC overview
- When should you use OIDC?
- Using OIDC instead of custom authentication
- OIDC Authorization Code
- Scopes
- User consent
- Token exchange
- Handle the ID token
- ID token structure
- Validating the signature
- Verifying the ID token
- Best practices with ID tokens

* Optional interactivity

Security Compass