ANG101 - DEFENDING ANGULAR

Course Learning Objectives

This course has three areas of focus for software developers. The first identifies security-related differences in Angular, and how to handle authentication, authorization, and OAuth 2.0.

The second area explains how to mitigate common injection vulnerabilities, configure and enforce browser-based defenses, and implement best practices for deploying HTTPS.

Finally, the third part focuses on secure authentication using forms or OIDC, propagating authorization states, and avoiding common pitfalls.

Description

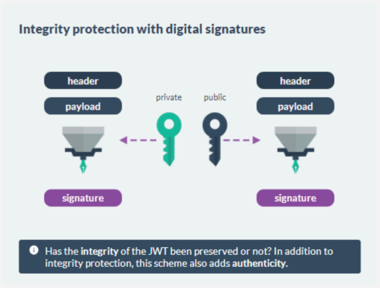
Defending Angular is divided into three parts. Part one helps software developers investigate how the Angular development paradigm impacts security. Part two explores a set of best practices for building, deploying, and maintaining Angular applications. And Part 3 investigates how to implement authentication and authorization in Angular applications.

Audience

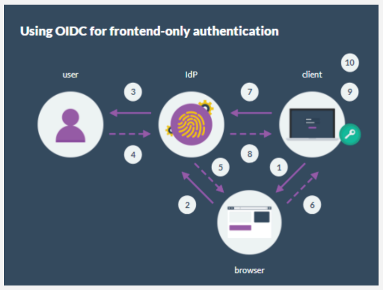
Angular developers

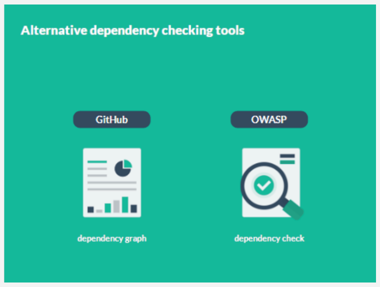
Time Required

Tailored learning - 120 minutes total (approx.)



Has the **integrity** of the JWT been preserved or not? In addition to integrity protection, this scheme also adds **authenticity.**





Security Compass

ANG101 - DEFENDING ANGULAR

Course Outline

1. Introduction to Angular security

- The server-side model
- The client-side model
- Overview of changes
- Client-side page composition
- Client-side XSS overview
- Page loads versus API calls
- Overview of authorization challenges
- Remote code versus dependencies
- The problem with NPM dependencies
- Addressing vulnerabilities in dependencies

2. Role of authentication and authorization

- Enabling authorization
- Authentication in practice
- Session management in practice
- Authorization in practice
- A modern authentication landscape

3. Enforcing authorization

- Authorization in Angular applications
- Authorization is a backend responsibility
- Introduction to authorization states
- Tracking authorization state
- Propagating authorization state
- Alternative solutions

4. OAuth 2.0

- The conceptual idea of OAuth 2.0
- Using OAuth 2.0 in Angular
- Challenges with OAuth 2.0
- The "backend-for-frontend" pattern

5. XSS

- XSS in Angular
- Refresher on XSS
- XSS in Angular applications
- Strict contextual escaping
- Sanitization
- Follow the Angular way
- Avoid bypassSecurity functions

6. Advanced XSS

- URL injection attacks
- URL injection attack example
- The problem with resource URLs
- Trusting resource URLs
- Template injection attacks
- Preventing template injection

7. Dependency management in Angular

- Vulnerabilities in dependencies
- Typosquatting attacks
- Compromised packages
- Targeted attacks
- Using NPM audit
- Alternative dependency checking tools
- Best practices for managing dependencies

8. HTTPS

- HTTPS for Angular applications
- Configuring HTTPS for the frontend
- Configuring HTTPS for the backend
- HTTP Strict Transport Security
- Securing your TLS configuration
- The certificate security ecosystem
- Certificate Authority Authorization
- Certificate Transparency

9. HTTP headers

- Header-based security policies
- Overview of policies
- Strict-Transport-Security
- X-Content-Type-Options
- X-Frame-Options
- Content-Security-Policy
- Latest developments in CSP
- Referrer-Policy
- Depreciated headers

SecurityCompass

ANG101 - DEFENDING ANGULAR

Course Outline

10. Form-based authentication and OpenID

- Simple authentication in Angular
- Supporting authentication in the backend
- A conceptual overview of OpenID Connect
- Using OIDC to centralize authentication
- Using OIDC for frontend-only authentication
- Choosing the right OIDC flow
- Pitfalls with implementing OIDC

11. Cookies, CSRF, and CORS

- Handling cookies
- Cookies are present on all requests
- Configuring cookies for HTTPS
- Preventing cookie access from JavaScript
- Avoid using domain-wide cookies
- Overview of current best practices
- Handling CSRF
- Examining the CSRF attack surface
- APIs must be strictly defined
- Using CORS as a CSRF defense
- Alternative CSRF defenses for APIs

12. Custom mechanisms

- Using a custom mechanism to propagate state
- Custom implementation with an Interceptor
- Restrictions on sending authorization state
- Overview of a custom state mechanism
- Using JSON Web Tokens
- Integrity protection with HMACs
- Integrity protection with digital signatures
- JWT library support
- Security-relevant JWT metadata
- JWTs are not session objects
- Using JWTs in a larger ecosystem