

# SWF201 - DEFENDING iOS (SWIFT)

## Course Learning Objectives

The OWASP Top 10 provides a list of common vulnerabilities in software application, and apps developed in iOS with Swift are no exception. This course details a baseline guidance for developers to address vulnerabilities in iOS apps by delving into the causes of common vulnerabilities and the defenses to mitigate them. Developers will explore secure coding practices that defend against weaknesses such as authentication, sensitive data leakage, and injection attacks, and Apple's proprietary security tools such as App Transport Security for secure data transfer, and Secure Enclave for key storage.

## Description

Explore defenses against common vulnerabilities in iOS applications developed with Swift.

This course covers industry best practices in secure coding as it relates to authentication and authorization, session management, secure data transfers, secure data storage, cryptography, and secure data ingestion.

### Audience

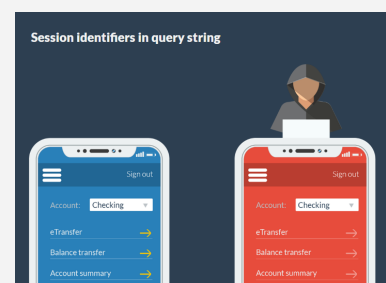
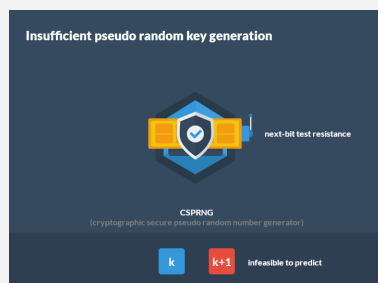


iOS mobile application developers  
iOS application architects  
Security professionals

### Time Required



Tailored learning - 70 minutes total



# SWF201 - DEFENDING iOS (SWIFT)

## Course Outline

### 1. Authentication and authorization

- Authentication vs authorization
- Untrusted incoming requests
- Client-side authentication bypass
- No account lockout or throttle
- Insufficient password policy requirements
- Insufficient authorization requirements
- Integration with password managers
- Password management applications
- Suggest strong password at account creation
- Defining textContentType
- Token management on the server side
- How OAuth works
- About 'appsecret\_proof'

### 4. Cryptography

- Insufficient pseudo random key generation
- Symmetric key cryptography with hardcoded keys
- Code: Random key generation
- Intro to Apple Secure Enclave
- Using Apple Secure Enclave
- Code: How does it work?
- Code: Generate key
- Code: Decrypt the cipher
- Managing keys, certificates, and passwords

### 2. Secure data transfer

- Unencrypted communications
- Improper certificate validation
- Code: Improper certificate validation
- Intercepting communication
- Using mitmproxy
- App Transport Security (ATS)
- Code: App Transport Security (ATS)
- Using CryptoKit
- Certificate pinning
- Code: NSURLConnectionDelegate
- Code: Disabling ATS

### 5. Secure data ingestion

- About secure data ingestion
- Client-side SQL Injection
- WKWebView input
- Keyboard data caching
- Third-party keyboards
- Parameterizing SQL statements
- Safer SQL solution
- Disable UITextField caching and indexing
- Validate user input
- Detect third-party keyboards
- iOS launch and environment constraints

### 3. Secure data storage

- Sensitive data stored in plaintext
- Sensitive data stored on a device
- Hardware-backed encryption
- Background apps
- Automatic snapshots
- Shared clipboards
- Screen recording and broadcasting
- Store sensitive data
- App data best practices
- Store data in the iOS keychain
- Code: Store data in the iOS keychain
- Code: Clear data for background apps
- Privilege escalation
- Sanitize the snapshot screen
- Private pasteboards
- Code: Sanitize content