IOS201 - DEFENDING iOS

Course Learning Objectives

The OWASP Top 10 provides a list of common vulnerabilities in software application, and apps developed in iOS with Objective-C and 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

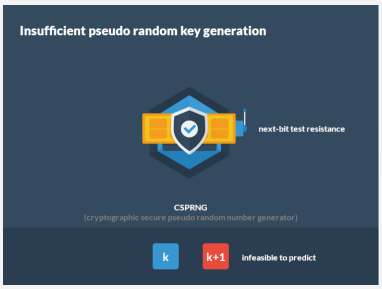
Explore defenses against common vulnerabilities in iOS applications developed with Objective-C and Swift. This course covers industry best practices in secure coding as it relates to authentication and authorization, session management, secure data transfers, secure data storage, cryptography, and secure data ingestion.

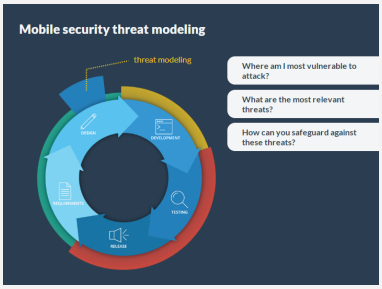
Audience

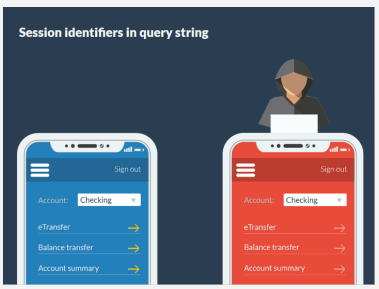
iOS mobile application developers
iOS application architects
Security professionals

Time Required

Tailored learning - 60 minutes total







IOS201 - DEFENDING iOS

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
- Token management on the server side
- How OAuth works
- About 'appsecret_proof'

2. Secure data transfer

- Unencrypted communications
- Improper certificate validation
- Code: Mismatched certificate bypass *
- App Transport Security (ATS)
- Using CryptoKit **
- Certificate pinning
- Code: NSURLConnectionDelegate
- Code: Pinning the TLS certificate *
- Code: Disabling ATS

3. Secure data storage

- Sensitive data stored in plaintext
- Sensitive data stored on a device
- Background apps
- Automatic snapshots
- Shared clipboards
- Screen recording and broadcasting
- Store sensitive data
- Store data in the iOS keychain
- Store data in the iOS keychain (example)
- Code: Keychain (Save Data) *
- Code: Keychain (Read Data) *
- Code: Keychain (Delete Data) *
- Clear data for background apps
- Sanitize the snapshot screen
- Private pasteboards
- Code: Private pasteboards
- Code: Sanitize content

4. Cryptography

- Insufficient pseudo random key generation
- Symmetric key cryptography with hardcoded keys
- Random key generation
- Using Apple Secure Enclave 1 of 3
- Using Apple Secure Enclave 2 of 3
- Using Apple Secure Enclave 3 of 3
- How does Secure Enclave work?
- Code: SecKeyCreateRandomKey
- Random key generation
- Code: Decrypt the cipher

5. Secure data ingestion

- About secure data ingestion
- Client-side SQL Injection
- WKWebView input 1 of 3
- WKWebView input 2 of 3
- WKWebView input 3 of 3
- Keyboard data caching
- Third-party keyboards
- Parameterizing SQLi commands
- Safer SQL solution
- Disable UITextField caching and indexing
- Validate user input
- Detect third-party keyboards

* Part of Objective-C only
** Part of Swift only

SecurityCompass