

iOS Application Penetration Testing

Static analysis, runtime testing and actionable reporting

FA-0642 · 5 days

Cybersecurity

Assess a mock iOS application using static and dynamic techniques, validate findings and produce a clear mitigation-focused report.

Introduction

This five-day course develops iOS application security-testing skills through a prepared mock application and isolated training devices or emulators. It covers architecture, static analysis, runtime behaviour, network traffic, instrumentation, vulnerability validation and reporting.

Exercises use explicitly authorised training targets and fictional data. Tool results need manual validation, and individual test coverage does not prove an application is free of vulnerabilities. The course is independent training: it does not claim a certification award, provider affiliation or guaranteed exam or professional success.

Learning outcomes

The course teaches participants to:

- Explain iOS components, permissions and relevant mobile security risks.
- Prepare a compatible iOS security-testing environment.
- Inspect an IPA, manifest and decompiled code for selected risks.
- Observe runtime behaviour, logs and network traffic in the mock application.
- Use selected instrumentation to evaluate a training app's jailbreak/integrity and certificate-pinning controls and their limitations.
- Validate storage, component, WebView, cryptographic and logic findings.

- Document reproducible evidence, impact, limitations and practical mitigations.
- Perform a scoped assessment of the supplied mock application.

Prerequisites

- Have a foundational understanding of mobile applications and iOS architecture.
- Be familiar with penetration testing concepts and methodologies.
- Possess basic knowledge of macOS commands and networking principles.

A prepared compatible macOS/Xcode toolchain, authorised mock-app build and agreed training simulator/device. Physical-device provisioning and any licensed analysis tools must be available in advance; a simulator does not reproduce every physical-device security property.

Training outline

1. Day 1 – iOS Security Foundations

Architecture and Permissions

- iOS OS structure: Kernel, Libraries, Frameworks, Applications
- iOS application lifecycle and components
- iOS permission model and security policies

Security and Assessment Context

- Key threats to mobile applications
- Overview of the OWASP Mobile Top 10 and how risk awareness relates to MASVS controls and MASTG test guidance; a Top 10 list is not a complete test plan.
- Secure software development lifecycle (SDLC) for iOS

Training Environment

- Inspect the prepared Xcode and simulator environment; match tools to the supplied mock-app build.
- Use an authorised compatible physical training device where required; simulator and physical-device behaviour differ.
- Inspect the prepared analysis toolset: Hopper or an appropriate supplied equivalent, a proxy such as Burp Suite, Frida, Objection and MobSF. Device/OS architecture and tool licensing affect availability.

2. Day 2 – Static Analysis

IPA Structure and Decompilation

- IPA structure and components
- Inspect disassembly and available metadata with a suitable analysis tool; class-dump is relevant to Objective-C metadata and does not reconstruct complete original Swift source.

Reverse Engineering

- Analyzing Info.plist and entitlements
- Identify sensitive-data exposure, entitlements and permission risks; distinguish necessary access from vulnerabilities.
- Introductory small inspection where useful; avoid claiming exhaustive reverse-engineering expertise.

Selected Static Risks

- Hardcoded sensitive data
- Misconfigured permissions and URL schemes
- Use of outdated libraries and components

3. Day 3 – Dynamic Analysis and Instrumentation

Runtime Behaviour

- Monitoring application behavior during execution
- Inspect available runtime logs using macOS Console or suitable supplied tooling; log availability varies by OS, build and device.

Traffic Analysis

- Setting up a proxy with Burp Suite
- Analyse traffic generated by the supplied mock application and its training endpoint.
- TLS/certificate-validation weaknesses and relevant app transport configuration.

Instrumentation

- Introduction to Frida and Objection

- Instrument and modify selected mock-app behaviour to test a hypothesis.
- Evaluate jailbreak-detection and certificate-pinning bypasses in the deliberately testable application; bypassability depends on implementation and environment and is not universal.

4. Day 4 — Vulnerability Validation

Storage and Platform Components

- Assess inappropriate sensitive-data storage in UserDefaults, Core Data and Keychain configuration/access groups. Keychain is a protected credential store, not inherently insecure; storage suitability and protection classes depend on the use case.
- Improper use of WebView and JavaScript bridges
- Assess URL-scheme and deep-link handling in the supplied mock app.

Logic and Runtime Behaviour

- Validate a selected logic flaw and document a bounded proof of concept on the mock app.
- Debugging applications for hidden behaviors

Security Misconfiguration

- Weak encryption and cryptography practices
- Debuggable apps in production

5. Day 5 — Reporting, Mitigation and Assessment

Evidence and Reporting

- Writing detailed and actionable reports
- Provide reproducible, minimal mock-app proofs of concept with evidence and limitations.

Mitigation Review

- Secure coding guidelines
- Implementing defense-in-depth strategies

Scoped Mock-App Assessment

- Scoped penetration-test exercise on the supplied mock application, not a guarantee of exhaustive coverage.
- Applying learned techniques to identify and report vulnerabilities

Review

- Review of key concepts
- Q&A and next learning steps; no certification award or exam-success promise.