OFFENSIVE MOBILE REVERSING AND EXPLOITATION

Expert-Led Cybersecurity Training · Beginner to Advanced

COURSE OVERVIEW

This comprehensive course offers an in-depth exploration of both iOS and Android operating systems, focusing on their internals and security features. The iOS segment of the course dives into the architecture of iOS, memory management, application sandboxing, code signing, and advanced mitigations like SPTM, TXM, PAC, PAN, and PPL. Students will also receive a thorough introduction to the ARM64 architecture, including static and dynamic analysis techniques, debugging tools, and disassembly tools. Moving into iOS application security, students will explore topics such as code signing, encryption, secure communication, and the use of Frida for dynamic instrumentation. Advanced topics like hooking, memory manipulation, and instrumenting network communication will also be covered. The course also covers iOS malware analysis, including static, dynamic, and behavioral analysis, along with mitigation and prevention strategies.

On the Android side, participants will gain a broad understanding of Android system architecture, including drivers, modules, the Linux kernel, and the Android Binder. Hands-on experience in reverse engineering, exploit development for ARM platforms, memory management, and vulnerabilities will be provided. The course also covers Android's boot, recovery, rooting processes, and permissions, along with security features like DAC, CAP, SECCOMP, and SELinux.

8KSEC

For a practical learning experience, the course covers how to extract and decrypt boot images for Android devices. The course covers handson exercises for symbolicating the Android kernel and porting exploits to other Android devices. Advanced Frida techniques such as custom tracing, profiling, and memory inspection are explored with real-world applications. Case studies on prominent malware and custom malware samples designed for the course shed light on reverse engineering and advanced forensics techniques. Application Security related vulnerabilities occurring due to Android components are also covered as a part of the course.

Throughout the course, participants will engage in practical labs to gain hands-on experience with iOS and Android internals, application security, reverse engineering, and vulnerability analysis. By the end of the course, students will have the skills needed to reverse engineer, design, develop, and secure iOS and Android applications effectively, as well as have a good understanding of all the security measures implemented in Android/iOS Userland and Kernel.

This course prepares you for the **Offensive Mobile Security Expert (OMSE)** certification exam, a hands-on assessment specifically designed to test your grasp of advanced mobile security domains including userland and kernel components.

KEY LEARNING OBJECTIVES

- Get an understanding of the latest ARM64 instruction set
- Learn the internals of Mobile Kernels along with several Kernel security mitigations
- Learn Device Fingerprinting and Anti-Fraud techniques
- Advanced Dynamic Instrumentation using Frida
- Understand some of the latest bugs and mitigations (PAC, CoreTrust, PPL, etc)
- Get an intro to common bug categories like UaF, Heap overflow and more
- Understanding how Rooting and Jailbreaks work
- Reverse engineer iOS and Android binaries (Apps and system binaries)
- Learn how to audit iOS and Android apps for security vulnerabilities
- Understand and bypass anti-debugging and obfuscation techniques
- Get a quick walkthrough on using Ghidra, radare2, Hopper, Frida and other tools
- Learn how accessibility malwares work, and how to reverse engineer well-known crypto wallet stealers
- Learn how to symbolicate the iOS and Android kernel
- Learn how to extract and decrypt boot images for Android devices
- Perform patch diffing on iOS updates to spot security-relevant code changes
- Extract and prepare binaries, then use key tools for analysis
- Reverse engineer and trace Android JNI bindings, including RegisterNatives
- Use tools like JNINinja, Frida, and Medusa to hook and monitor JNI methods at runtime
- Build a functional JVM environment and AFL++ Frida-mode to fuzz JNI bindings and validate crashes
- Become an Offensive Mobile Security Expert (OMSE)

WHY SHOULD YOU TAKE THIS COURSE?

This is a completely hands-on course designed for beginners and intermediate students. Instead of just slides, attendees will get a chance to exploit all of the vulnerabilities taught by the instructors. For the On-site and Virtual sessions, the attendees will be provided with Cloud-based Corellium labs for performing the hands-on iOS and Android exercises without the need to carry physical phones. A Slack channel is created before the course for the students so that they can be adequately prepared in terms of hardware and software before the class.

HARDWARE/SOFTWARE REQUIREMENT

- Laptop with: 8+ GB RAM and 40 GB hard disk space
- Students will be provided with access to Linux cloud instances
- Students will be provided with access to Corellium for iOS and Android hands-on and as such do not need to carry physical devices
- Administrative access on the system

Detailed Course Setup instructions and Slack access will be sent a few weeks prior to the class

PREREQUISITE KNOWLEDGE

- Working knowledge of cybersecurity and pentesting fundamentals
- Basic working knowledge of iOS and Android platforms
- Basic Linux skills and command-line proficiency
- Understanding of fundamental programming concepts and looping structures in at least one higher-level language (Java, Kotlin, Objective-C, Swift, C, C++, or similar)
- Basic ARM/AARCH64 binary assembly and exploitation knowledge is recommended, but not required

WHO SHOULD ATTEND?

This course is for penetration testers, mobile developers or anyone keen to learn mobile application security and wants to get started in OS exploitation.

WHAT WILL THE STUDENTS GET

- An attempt to Offensive Mobile Security Expert (OMSE) certification exam
- Certificate of completion for the Training program
- Source code for vulnerable binaries used during the class
- Source code for Exploit PoCs' that can be used for Bug Bounties
- All Python Scripts used during the course
- Students will be provided with access to Corellium for the duration of the course
- Students will be provided access to cloud instances for the duration of the course
- Slack access for the class and after for regular mobile security discussions

8KSEC

COURSE SYLLABUS

Module 1: Introduction to Reverse Engineering in iOS and Android

- Key Concepts and Terminologies
- Introduction to Hopper/Ghidra
- Introduction to the ARM 64 instruction set
- ARM64 security mitigations
- ARM64 calling convention
- Introduction to Objective-C and Swift
- Reversing Objective-C and Swift Binaries
- Introduction to Java and Kotlin
- Disassembling methods
- Modifying assembly instructions
- Deciphering Mangled Swift Symbols
- Identifying Native Code
- Understanding the Program flow
- Identifying Cross-Platform mobile frameworks
- Reversing ARM binaries
- Exploiting a simple Heap Overflow
- Building a simple ROP chain
- Breaking ASLR with Info leaks/Brute force
- Exploit mitigations (ASLR, Heap Poisoning, PAN, etc)

Module 2: Getting Started with iOS Security

- iOS security model
- App Signing, Sandboxing, and Provisioning
- iOS App Groups
- Primer to iOS 17-18 security
- Xcode Primer
- Address Sanitizer
- Exploring the iOS filesystem
- What's in a Code Signature?
- Entitlements explained
- How Sandboxing works on iOS
- Setting up lldb for Debugging
- lldb basic and advanced usage
- Setting up the testing environment
- Jailbreaking your device
- What's in a Rootless Jailbreak?
- Jailbreak Bootstraps
- Sideloading apps
- Binary protection measures
- Decrypting IPA files
- Self-signing iOS binaries
- Analyzing Proprietary security Mitigations
- Overview of Past Vulnerabilities
- Intro to dyld_shared_cache

Module 3: iOS Kernel internals

- Intro to XNU kernel
- The Mach and BSD Layer
- Overview of IOKit
- Extracting the Kernelcache and Kexts
- Analyzing specific kexts AMFI, CoreTrust, Sandbox
- Sandbox Profiles
- Symbolicating iOS Kernelcache
- Overview of mach_msg2, SAD_FENG_SHUI, PGX
- Entitlement validation in the Kernel
- Analyzing Kernel Panic files
- Walkthrough of PAC, SPTM, PAN, GXL, PPL etc
- Patching Diffing XNU kernel

(Continued on the next page)

Module 4: Frida in-depth

- Overview of Frida and its capabilities
- Setting up the Frida environment
- Frida usage and commands
- Frida-trace and handlers
- Frida hooking techniques
- Frida on Swift applications
- Frida on native code
- Frida memory manipulation techniques
- Analyzing messaging apps using Frida
- Invoking custom functions with Frida

Module 5: iOS application vulnerabilities

- Tracing Crypto operations
- Side channel data leakage
- Sensitive information disclosure
- Bypassing Jailbreak Detection
- Bypassing SSL Pinning
- Bypassing Certificate transparency checks
- Exploiting iOS WebViews
- Exploiting URL schemes and Universal LInks
- Client-side injection
- Bypassing jailbreak, piracy checks
- Inspecting Network traffic
- Traffic interception over HTTP, HTTPs
- Manipulating network traffic
- Identifying iOS malware

Module 6: iOS vulnerabilities

- Case Study of Sandbox Escapes
- Incorrect validation of Entitlements
- XPC Related vulnerabilities
- Case Study of a Kernel Vulnerability
- Case Study of a PAC Bypass

Module 7: iOS Malware Reversing

- Understanding different stages of a Malware
- Device Acquisition techniques
- Using Custom IOCs
- Case Study of some Public Malware

Module 8: Securing iOS Ecosystem

- AppAttest and Device Check frameworks
- Device Fingerprinting
- Detecting GPS Spoofing
- Implementing Secure Webviews
- Code Obfuscation techniques
- Protecting the Transport Layer
- Detecting Malicious Libraries
- Implementing Anti-Debug Checks
- Detecting Suspicious Device Reset
- Detecting Patched Applications
- Detecting Proxied Applications
- Jailbreak Detection Techniques
- Pasteboard Security Measures
- Understanding the Lockdown Mode
- Understanding Code Signature Checks

Module 9: Intro to Android Security

- Android Security Architecture
- Extracting APK files from Google Play
- Understanding Android application structure
- Signing Android applications
- Understanding Android ADB
- Understanding the Android file system
- Permission Model Flaws
- Attack Surfaces for Android applications

(Continued on the next page)

Module 10: Android Components

- Understanding Android Components
- Introducing Android Emulator
- Introducing Android AVD
- Setting up Android Pentest Environment

Module 11: Reversing Android apps

- Process of Android Apps Engineering
- Reverse Engineering for Android Apps
- Smali Learning Labs
- Examining Smali files
- Dex Analysis and Obfuscation
- Reversing Obfuscated Android Applications
- Exploiting Android Accessibility Permissions
- Reverse Engineering known complex Malwares in the Wild
- Patching Android Applications
- Android App Hooking

Module 12: Static and Dynamic analysis

- Proxying Android Traffic
- Exploiting Local Storage
- Exploiting Weak Cryptography
- Exploiting Side Channel Data Leakage
- Exploiting Content Provider Path Traversal & Info Leakage
- Multiple Manual and Automated Root Detection and Bypass Techniques
- Exploiting Weak Authorization mechanism
- Identifying and Exploiting Android Components
- Exploiting Android NDK
- Android Game Hacking
- Multiple Manual and Automated SSL Pinning Bypass techniques
- Firebase Exploitation
- Exploiting Biometric Authentication
- In-memory tampering
- Exploiting Flutter Applications
- Exploiting AWS Cognito Misconfiguration
- Exploiting Android Deep Links and WebViews

Module 13: Frida and Automated Exploitation

- Exploiting Crypto using Frida
- Basic App Exploitation techniques using Frida
- Dumping Class Information using Frida
- Dumping Method Information using Frida
- Viewing and Changing Information using Frida
- Calling Arbitrary functions using Frida
- Tracing using Frida
- Advanced App Exploitation techniques using Frida
- Frida on non-rooted Android

Module 14: Securing Android Apps

- Detecting Patched Android Applications
- App Integrity Protection
- Detecting Malicious Libraries
- Detecting Emulator/Rooted Devices
- Secure Implementation of WebViews
- Implementing Anti-Debug Checks
- Detecting Suspicious Device Reset
- Detecting Proxied Applications

(Continued on the next page)

Module 15: Android Kernel

- Android Boot process and Bootloader interaction
- Customizing and Building Android Kernel for Vulnerability Research
- Android Rooting Process
- Debugging Android Kernel and binaries
- Extract Android kernel from Boot image
- Symbolicating the Android Kernel
- Privilege Escalation on Android
- SELinux explained
- Overview of Kernel protections and bypasses

About the company

8kSec is a foremost cyber security research company offering exceptional training and consulting services to aid clients in enhancing their security stance. Our experts possess extensive experience in delivering specialised cybersecurity training and consulting to several commercial and defence organisations across the United States, Europe, and the Middle East and North Africa region.

Get in touch

8kSec.io
info@8ksec.io