CORELLIUM

iOS Reverse Engineering

Course Goal: To provide students with the fundamental skills of iOS reverse engineering (dynamic and static analysis) and using those skills to explore the Nickname vulnerability.

Day 1: Foundations & Dynamic Instrumentation

Objective: Understand the iOS security model and gain initial access to our target process, imagent, using dynamic instrumentation.

Module 1: The Modern iOS Security Landscape

- **Architecture & Security Primer:** The iOS Sandbox, Code Signing, Entitlements, and the role of key system processes.
- **Introduction to the Target:** What is imagent? Why is it a critical target for researchers?
- **Setting Up the Lab:** Introduction to Corellium and connecting to a virtualized iOS device.

Module 2: Introduction to Dynamic Instrumentation with Frida

- **Frida Fundamentals:** Hooking functions, tracing method calls, and modifying behavior at runtime.
- **The Objective-C Runtime:** A primer on how to target Objective-C methods and classes with Frida.

Lab 1: Dynamic Instrumentation with Frida

- Use Frida to attach to the imagent process on a Corellium device.
- Trace Objective-C method calls within the IMDNicknameController class to observe its activity in real-time.
- Modify a simple return value from a function within imagent.

Day 2: Static Analysis & Debugging

Objective: Learn to analyze compiled code, triage a crash log, and inspect a process with a debugger.

Module 3: Static Analysis with Ghidra/IDA Pro

- **Navigating the Binary:** Loading imagent into a disassembler.
- **Identifying Code Paths:** Finding key functions, cross-references, and Objective-C class structures.

Module 4: Demystifying Crashes with LLDB

- **Reading an iOS Crash Log:** Exception types, thread backtraces, and binary images.
- **Symbolication & ASLR:** Understanding why addresses in a crash log don't match your
- disassembler and how to fix them.
- **Introduction to LLDB for iOS:**
  - Attaching to a running process and essential Commands.
  - Commercial in Confidence TFP0 Labs, Inc
  - Breakpoints and Watchpoints.
  - Memory Examination.
  - Runtime Manipulation.
  - Thread Analysis.

Lab 2: Dissecting a crash

- Students will be provided with a sample, unsymbolicated crash log. They will practice "unsliding" addresses and manually symbolicating the backtrace using Ghidra/IDA Pro.
- **Hands-on Debugging:** Students will then trigger a pre-configured crash on their device and attach LLDB to print the backtrace, examine the CPU registers, and inspect the memory addresses involved in the crash, correlating their findings with the crash log.

Day 3: CoreFoundation and IPC

Objective: Introduction to fundamental Apple frameworks.

Module 5: CoreFoundation & Objective-C Object Model Intro

- **Toll-Free Bridging:** Understanding the direct relationship between CoreFoundation types (CFTypeRef) and Objective-C objects.

- **Memory Management in Practice:**
  - Automatic Reference Counting (ARC) vs. Manual Reference Counting (MRC).
  - **Dynamic Observation:** Using Frida to inspect an object's retain count at runtime to see memory management in action.
- **Mutable vs. Immutable Collections:**
  - A detailed examination of NSDictionary vs. NSMutableDictionary.
  - **Thread-Safety Implications:** Discussing why mutable collections are not thread-safe.
  - **Common Mitigation Patterns:** Introducing developer patterns for protecting mutable collections.

Module 6: Inter-Process Communication (XPC)

- **How iOS Processes Talk:** An introduction to XPC, the primary IPC mechanism on iOS.
- **Serializing Objects:** How objects like NSDictionary are packaged and sent over an XPC connection.
- **Tracing XPC:** Using Frida to intercept and inspect XPC messages.

Lab 3: Reverse Engineering Nickname's IPC and Object Handling

- Students will write Frida scripts to intercept XPC messages being sent from imagent.
- They will trace the methods in IMDNicknameController that handle incoming nickname data.

Day 4: Exploring the "Nickname" Vulnerability

Objective: Synthesize all learned skills to perform a full analysis of the "Nickname" vulnerability and analyze Apple's fix.

Module 7: The "Nickname" Vulnerability: Root Cause, Patch Analysis & Reproduction

- Detailed walkthrough of the race condition in IMDNicknameController's _broadcastNicknamesMapChanged method.
- **Connecting the Dots:** Linking Day 2's LLDB analysis with Day 3's understanding of mutable objects and XPC serialization.
- How to effectively use binary diffing tools on patched binaries.

Lab 4, Part 1 (Patch Analysis)

Students will be provided with vulnerable and patched versions of the IMDaemonCore binary. They will use a binary diffing tool to identify the exact code changes made by Apple.

Lab 4, Part 2 (Crash Reproduction)

- Students will leverage their knowledge from previous days to craft a specific sequence of actions on a vulnerable Corellium device to trigger and reproduce the imagent crash.
- Capture and analyze the CoreTrace log generated during the crash to find evidence of thread contention.

Course Prerequisites

The target audience for this training consists of beginner-intermediate level security researchers and developers interested in iOS internals and runtime analysis. Students should have:

- Basic understanding of operating systems architecture (userland/kernel separation)
- Basic familiarity with debuggers (LLDB/GDB basic commands)
- Working knowledge of C (Objective-C experience would be helpful)
- Basic JavaScript skills (for Frida)
- Understanding of process memory layout (heap, stack, registers)

CORELLIUM®