Antid0te SG

macOS & iOS Kernel Internals for Security Researchers

Low-level kernel internals for vulnerability analysis, debugging, and modern mitigations on macOS Tahoe and iOS 26.

At a glance

Item	Details
Instructor	Stefan Esser
Format	Live online (Zoom). 5 hours lecture/day plus tasks. Next day starts with solution discussion.
Duration	5 days
Audience	Security researchers, reverse engineers, security engineers
Editions	EU/NA and APAC timezones
Recordings	Available for a limited time after the training

Pricing

Currency	Price
EUR	4500,- EUR
SGD	6500,- SGD
USD	5000,- USD

We usually charge in SGD, but can charge in EUR and USD if requested.

Schedule across time zones

EU / North America edition (live lecture block)

17:00–22:00	Berlin
16:00–21:00	London
08:00–13:00	Seattle / Vancouver
11:00–16:00	New York / Montreal
23:00–04:00 (+1)	Singapore
01:00–06:00 (+1)	Sydney

APAC edition (live lecture blocks)

10:00–12:30 / 13:30–16:00	Singapore (SGT)
09:00–11:30 / 12:30–15:00	Bangkok / Jakarta
11:00–13:30 / 14:30–17:00	Tokyo / Seoul
12:00–14:30 / 15:30–18:00	Sydney / Melbourne
07:30–10:00 / 11:00–13:30	India
14:00–16:30 / 17:30–20:00	Auckland / Wellington

Overview

With macOS Tahoe and iOS 26 Apple continues to harden the platform with ARM64 hardware-assisted security mitigations and additional security boundaries.

This training focuses on kernel internals from a security researcher’s perspective: vulnerability analysis, crash triage, driver attack surface, and understanding mitigations in practice.

The course is lecture-driven with daily hands-on tasks. After each session you work on exercises, and the next day starts with solution discussion and common pitfalls.

Prerequisites

• Comfortable with C; basic Python helps for tooling and scripts

• Basic x86_64 and/or ARM64 assembly familiarity

• Reverse engineering experience (IDA, Ghidra, Binary Ninja, etc.)

• Recommended: Apple Silicon macOS device for the ARM64-focused exercises

What you’ll be able to do

• Understand how XNU is structured (Mach, BSD, IOKit) and how subsystems fit together

• Apply a practical approach for crash triage and root cause analysis

• Reason about IOKit, DriverKit/SystemExtensions, and EndpointSecurity from an attack-surface perspective

• Understand VM/page tables/permissions and the security implications on Apple platforms

• Understand Apple proprietary mitigations (including PPL/GXF and related mechanisms) and how the kernel uses them in practice

Day-by-day outline

Day 1 — Kernel boot chain and ARM64 foundations

Get a solid introduction to what the kernel is and where to find it in binary and source code form. Learn how the source tree is structured and what you can find where. Build the foundations needed to understand ARM64 security mitigations (including Apple-specific mechanisms). Learn how to collect kernel core dumps and how to attach a debugger to the kernel for practical analysis.

- Kernel binary and firmware
- Kernel source code
- Booting a custom kernel
- Kernel panic dumps and KDP
- ARM64 internals (incl. Apple proprietary features and registers)
- KASAN on ARM64

Day 2 — Mach IPC and kernel security frameworks

Learn Mach messages and ports, two key concepts in the Mach part of XNU. Learn about kernel extensions, what they are used for, and how to write your own. Then learn several kernel-extension level security frameworks and how they are used to control or restrict behavior.

- Mach ports and Mach messages
- Mach Msg 2 (mach_msg2)
- GXF and PPL
- Kernel extensions
- Kauth
- IP filters
- Socket filters
- MAC Framework (Part 1)

Day 3 — MAC policies and newer security components

Understand how the MAC Framework powers code signing, entitlement handling, and the sandbox, and how these pieces fit together in practice. Learn the code signing monitor interface and the newer security components SPTM and TXM. Then dive into IOKit internals and the driver attack surface.

- MAC Framework (Part 2)
- Code Signing Monitor
- Secure Page Table Monitor (SPTM)
- Trusted Execution Monitor (TXM)
- IOKit introduction

Day 4 — Kernel heap internals and hardening

Understand the kernel heap in depth, including how it works and how it is hardened against attackers and advanced heap manipulation techniques. Learn about read-only kernel data structures and how they are used to thwart common attacks. Understand how MTE is used in the kernel heap.

- Kernel heap
- Zone element hardening
- kalloc heaps and kalloc types
- Read-only kernel data structures
- Zone allocator MTE
- Kernel address obfuscation
- Mach port hardening

Day 5 — Exclaves ecosystem, system extensions and audit targets

Get an overview of the Secure Kernel (SK) and Exclaves, including what problems they are designed to solve and how they relate to the rest of the platform. Learn what System Extensions are and how they function internally, with a focus on Endpoint Security extensions. Cover remaining kernel-level mitigations that were not discussed earlier. Finish with a guided look at other interesting areas as audit targets.

- Secure Kernel (SK)
- Exclaves
- System Extensions
- DriverKit internals
- Endpoint Security extensions
- macOS and iOS kernel-level mitigations
- Audit targets inside the kernel