Antid0te SG

Deep Dive into SPTM, TXM, SK and Exclaves

A deep technical dive into Apple’s newer security components, with practical analysis and research tooling using the GLx Research Platform.

At a glance

Item	Details
Instructor	Stefan Esser
Format	Live online (Zoom) with guided analysis and tasks between sessions
Duration	5 days (daily live sessions ~5h plus break)
Audience	Security researchers with prior macOS/iOS kernel foundations
Tooling	GLx Research Platform used for controlled execution, debugger attachment and introspection
Support	Discord support channel; recordings available for a limited time

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 expand platform security with ARM64 hardware-assisted mitigations and additional security boundaries. Components such as SPTM and TXM have been rolled out to more devices, and newer platforms also include the Secure Kernel (SK) and Exclaves.

Unlike our Kernel Internals training, this course focuses on these newer non-kernel security components. The goal is a research-grade understanding of what each component does, how the bootstrapping sequence progresses, how data moves across boundaries, and where the interesting audit angles are.

Hands-on analysis is supported by the GLx Research Platform. We use it to run SPTM, TXM, SK and Exclaves inside a controlled hypervisor environment and attach debuggers with full introspection of state, memory and control flow. Where applicable, we also cover running SPTM and TXM in userland using binary instrumentation. We additionally cover code coverage analysis and fuzzing approaches to evaluate interfaces and validation logic.

Prerequisites

• Solid reverse engineering foundations (IDA/Ghidra/Binary Ninja)

• Comfortable with C; basic Python helpful for tooling/scripts

• ARM64 assembly familiarity

• Prior macOS/iOS kernel foundations strongly recommended (our Kernel Internals course is a good precursor)

What you'll be able to do

• Understand the roles, boundaries, and integration points of SPTM, TXM, SK and Exclaves, including how XNU interacts with them

• Run these components in a controlled hypervisor environment using GLx Research Platform and attach debuggers with full introspection

• Analyze Tightbeam message formats, validation boundaries, and trust decisions at component interfaces

• Use code coverage and fuzzing approaches where applicable to evaluate component behavior and validation logic

• Evaluate mitigations and security mechanisms used by SPTM, TXM, SK and Exclaves, including TXM as the Code Signing Monitor implementation

Day-by-day outline

Day 1 — Guarded World overview and foundations

Welcome to the Guarded World. Understand what we are dealing with at binary level and whether there are even snippets of source code available that we can start our reverse engineering from. Get a good understanding of low-level ARM64 and Apple Silicon specific internals that are used throughout the components. Understand Apple DeviceTrees.

- Introduction to the Guarded World
- Binary view of SPTM, TXM, SK and Exclaves
- Source code view of SPTM, TXM, SK and Exclaves
- Local and remote kernel coredumps
- Apple DeviceTree
- ARM64 and Apple Silicon internals

Day 2 — SPTM fundamentals and GLx Research Platform basics

Understand the physical memory layout at boot time for SPTM. Get an introduction to the GLx Research Platform and how it can help us. Get a first glimpse at the SPTM binary and reverse engineer key points like GLx and exception handlers. Learn about the SPTM state machine. Get an idea what it takes to run SPTM in userland.

- Understanding the physical memory layout
- Introduction to the GLx Research Platform
- Getting started with the SPTM binary
- Learning about core data types inside SPTM
- GLx and exception handlers
- Frame types and memory typing
- The SPTM state machine
- Discussion of running SPTM in userland and the GLx platform

Day 3 — SPTM to TXM handoff, TXM, and Code Signing Monitor

Finish up the SPTM topic. Understand the handoff from SPTM to TXM. Understand the TXM bootstrapping process and the Code Signing Monitor it installs. Observe this in userland and inside the GLx platform. Learn how to do code coverage guided fuzzing of the code signing monitor inside TXM.

- Experimenting with SPTM in the GLx Research Platform
- Collect and visualize code coverage inside SPTM
- Understanding the SPTM to TXM handoff
- Getting started with the TXM binary
- Learning about core data types inside TXM
- Deep dive into the Code Signing Monitor API handler
- Discussion of running TXM in userland and the GLx platform
- Code coverage guided fuzzing of TXM code signing checks

Day 4 — Secure Kernel (SK/CL4) internals

Get a deep understanding of what the Secure Kernel is, how it works internally, and what API it provides to the Exclaves userland. Understand the core data types involved and the internal CL4 object types and system calls.

- Understanding the SPTM to Secure Kernel (SK/CL4) handoff
- Getting started with the Secure Kernel binary
- Learning about core data types inside SK
- Deep dive into the SK bootstrap
- Understanding CL4 object types
- Understanding the CL4 system calls
- Using the GLx platform to introspect the Secure Kernel

Day 5 — Exclave userland, scheduler, and instrumentation

Get a deep understanding of what is going on inside Exclave userland apps and how to communicate with the scheduler via the XRT protocol. Learn how this is used to encapsulate Tightbeam messages to directly communicate with Exclave components. Learn about upcalls and downcalls. Understand the full Exclave boot process, both within the Exclave world and on the XNU side. Learn about address space separation and how ExclaveKit fits into all of this. Learn all of this with full introspection available.

- Understanding the SK to ROOTTASK (Exclave userland) handoff
- Understanding Exclave threads and address spaces
- Dive into the Exclave scheduler XRT protocol
- Exclave communication via Tightbeam endpoints
- Exclave upcalls and downcalls
- The Exclave boot process
- Bringing up ExclaveKit
- Exclave resources and XNU
- Runtime instrumentation of Exclave apps