Agentic Deep Dive into SPTM, TXM, SK and Exclaves

Five-day in-person Singapore deep dive into Apple's newer security components, pairing the full Deep Dive curriculum with agentic reverse engineering workflows and GLx-based validation.

At a glance

Item	Details
Instructor	Stefan Esser
Format	In person in Singapore with lecture blocks, guided labs, and collaborative reversing
Dates	24-28 August 2026
Daily schedule	09:00-17:00 Singapore (SGT)
Duration	5 full days
Audience	Security researchers with prior macOS/iOS kernel foundations
Tooling	GLx Research Platform, debugger attachment, coverage and fuzzing workflows, AI-agent tooling where useful

Pricing

Currency	Price
EUR	5200,- EUR
SGD	7500,- SGD
USD	5990,- USD

We can invoice in EUR, SGD, or USD. Payment is possible via international bank transfer or Stripe credit card payment.

Training syllabus

Overview

This five-day in-person Singapore edition keeps the full technical scope of the Deep Dive training. The course covers SPTM, TXM, Secure Kernel, and Exclaves in depth: how the components fit together, where trust boundaries sit, how XNU interacts with them, and which areas matter most from a security-research and exploitation perspective.

The agentic angle is the differentiator, not a replacement for the core content. We look at where AI agents can genuinely help with diffing new versions, exploring unfamiliar binaries, recovering interfaces, summarizing changes, generating helpers, and proposing hypotheses while the human researcher remains responsible for direction, trust decisions, and validation.

A key part of the course is that the GLx Research Platform becomes part of the workflow. Selected GLx capabilities can be exposed to an AI agent via MCP-style interfaces or comparable tool bridges so the agent can support controlled runs, state inspection, trace queries, and iterative validation directly against the execution environment.

Prerequisites

• Solid reverse engineering foundations (IDA, Ghidra, or Binary Ninja)

• Comfortable with C and Python for tooling and experimentation

• Basic ARM64 assembly knowledge

• Prior macOS/iOS kernel internals experience is strongly recommended

• Prior AI-agent or MCP experience is helpful, but not required

What you'll be able to do

• Understand the roles, boundaries, and integration points of SPTM, TXM, Secure Kernel, and Exclaves, including XNU-facing interfaces

• Use GLx Research Platform for controlled execution, debugger attachment, state inspection, fault analysis, and exploit-path validation

• Analyze Tightbeam message formats, endpoint structure, transport behavior, and validation boundaries

• Reason about exploitation paths for SPTM, TXM, SK, and Exclaves with and without a compromised kernel

• Use agentic workflows to accelerate diffing, interface recovery, code navigation, triage, and hypothesis generation without giving up researcher control

• Evaluate where MCP-style or similar tool bridges make AI-agent access to GLx auditable and practically useful

Training syllabus

Day-by-day outline

Day 1 — Introduction, ARM64 foundations, and research environment

Build the architectural context for SPTM, TXM, Secure Kernel, and Exclaves research, then get comfortable with the controlled execution and introspection workflow used throughout the course.

• ARM64 foundations relevant to SPTM/TXM/SK/Exclaves research (system registers, page tables, exception handling)

• Apple proprietary ARM64 security mechanisms used in this space (for example SPRR, CTRR, and related features)

• Bird's-eye view: where SPTM, TXM, Secure Kernel, and Exclaves fit in the platform security architecture

• Using GLx Research Platform to run the components inside a controlled hypervisor environment

• Debugger attachment and full introspection across components (state, memory, control flow)

• Practical data sources: logs, memory maps, kernel coredumps, and correlation techniques

• Where agentic workflows help with environment setup, version diffing, and navigation of unfamiliar binaries

Day 2 — Tightbeam, XNU integration, and exploitation module

Understand the message and interface layer between XNU and the secure components, then use that context to reason about exploitability and mitigation boundaries.

• Tightbeam messages, endpoints, and transports

• Practical analysis of message formats and validation boundaries

• Communication between XNU and SPTM / TXM / Exclaves

• Key interfaces and boundaries: what data crosses and where trust decisions happen

• Typical audit angles: parsing, validation, state transitions, and error handling

• Exploitation paths for SPTM, TXM, SK, and Exclaves with and without a compromised kernel

• Building proof-of-concept exploits for demo vulnerabilities used in the training labs

• Using GLx to observe state transitions, fault paths, and exploit side effects during controlled runs

• Agentic support for diffing interface changes, clustering fault behavior, and summarizing experiment output

Day 3 — SPTM and TXM deep dive

Reverse engineer the lower secure-component layers, understand their data structures and handoff logic, and connect manual reversing with assisted exploration workflows.

• Reverse engineering SPTM and understanding its internal structure and key data types

• Bootstrapping and internal state transitions inside SPTM

• Running SPTM in userland with binary instrumentation

• Discussion of mitigations and security mechanisms used by SPTM

• Reverse engineering TXM and understanding its internal structure and key data types

• Evaluation of the Code Signing Monitor implementation

Training syllabus

• Running TXM in userland with binary instrumentation

• Discussion of mitigations and security mechanisms used by TXM

• Agentic assistance for code navigation, diff summarization, helper generation, and experiment planning

Day 4 — Secure Kernel internals

Build a strong model of how the Secure Kernel works internally, how its L4-style abstractions behave, and how to analyze exception paths and mitigations with full context.

• Reverse engineering SK and understanding its internal structure and key data types

• L4 system calls and L4 object types

• Debugging and introspection in the hypervisor environment

• Analyzing crashes and exception paths with full context (registers, state, memory)

• Discussion of mitigations and security mechanisms used by SK

• Agent-supported crash triage, call-path summarization, and validation of hypotheses against live debugger state

Day 5 — Exclaves and agentic GLx orchestration

Tie together ExclaveOS, ExclaveKit, scheduler-level behavior, and direct GLx-driven experimentation, including how agent tooling can be made useful without losing auditability.

• Reverse engineering ExclaveOS, ExclaveKit, and related components

• Debugging and introspection for Exclave apps in the hypervisor environment

• Runtime enumeration of components, ASIDs, and threads and correlating them with observed execution paths

• Discussion of mitigations and security mechanisms used by Exclaves / ExclaveOS

• Selected GLx capabilities exposed to an AI agent via MCP-style or comparable interfaces

• Practical patterns for using agents to drive controlled runs, query traces, correlate observations, and keep the researcher in control