Name of the Training

Offensive Agentic Android Security - Autonomous Exploit Generation and Variant Analysis in AI Era

Course Blurb

This course equips the students with the skills to leverage AI as an offensive Android security powerhouse. Students will build multi-agentic pipelines that decompose CVEs, generate exploit payloads, verify exploitation on real devices, and discover 0-day variants - all with minimal human intervention.

Starting from prompt engineering fundamentals and working up to fully orchestrated exploit lifecycles, the course covers the practical architecture behind agentic security workflows: Retrieval Augmented Generation (RAG) for ingesting vulnerability data at scale, Model Context Protocol (MCP) servers for real-time tool interaction, deterministic feedback loops for self-healing exploit generation, and delta-detection systems that automatically adapt to new patches and defenses.

Every technique is grounded in hands-on labs — from bypassing root detection and SSL pinning through automated repackaging and Frida instrumentation, to reproducing high-severity CVEs against Android system apps and generating variant payloads that circumvent incomplete fixes. Students will walk away with working pipelines, not slide decks.

Whether you're a mobile penetration tester, vulnerability researcher, or red teamer, this course gives you the architectural patterns and technical depth to build AI systems that don't just find bugs — they exploit them, verify the results, and hunt for the next one.

Description

In February 2026, security researchers uncovered Android malware that screenshots the infected device, feeds the image to Google's Gemini, and uses the AI's interpretation to simulate precise UI interactions — allowing it to persist without user awareness. This isn't a research prototype. It's commodity malware, in the wild, using AI as a core operational component.

The adversaries have moved. The question is whether you have too.

For most mobile security professionals, the daily reality hasn't changed much in years: manually reversing APKs, staring at decompiled Java and stripped native code for hours, writing and rewriting Frida scripts through tedious deploy-test-fail cycles, context-switching across a sprawl of disconnected tools, and hand-crafting proof-of-concept apps just to confirm whether a finding is actually exploitable. It works — slowly, painfully, and at a scale of one.

Meanwhile, Android's attack surface keeps expanding. Google publishes monthly security bulletins patching dozens of vulnerabilities. Third-party SDKs ship opaque code into millions of apps. System-level components carry privileges that a single logic bug can unlock. The volume of work that matters has outpaced what manual effort can reach.

This course exists to close that gap — not by adding AI as a novelty, but by rebuilding the offensive workflow around it from the ground up.

Throughout the training, attendees will learn to:

- Architect multi-agentic pipelines that chain LLM reasoning with deterministic tool execution for offensive Android security workflows
- Build and query RAG-backed knowledge systems that ingest CVE advisories, security bulletins, and decompiled source code at scale
- Construct self-healing exploit pipelines that generate payloads, deploy to real devices, verify exploitation, and autonomously reconstruct on failure
- Analyse Android attack surfaces systematically using AI — exported components, WebViews, deep links, content providers, and device logs
- Bypass root detection and SSL pinning through automated analysis of detection logic, strategy selection, and instrumentation code generation
- Trace vulnerability chains from third-party SDKs to host applications and score supply chain risk
- Integrate native code analysis tools like Radare2 and Ghidra through MCP servers into agentic workflows
- Diff pre-patch and post-patch AOSP commits to identify incomplete fixes and generate variant payloads
- Orchestrate physical devices for automated app lifecycle management — installation, instrumentation, log capture, and exploit verification
- Design self-learning loops that detect deltas in patched code and automatically update vulnerability knowledge bases

Every module features labs or case studies drawn from actual real world CVE's within core Android framework to give you practical experience:

Upon completion of this training, participants will be able to:

- Build end-to-end autonomous pipelines that take a CVE identifier as input and produce a verified, working exploit as output
- Reproduce high-severity Android CVEs from 2023–2025 through AI-driven analysis, PoC generation, and headless device verification
- Perform variant analysis using known vulnerabilities as seeds to discover unreported issues in privileged system components
- Evaluate and bypass common defense-in-depth mechanisms — root detection and SSL pinning — using both Frida-based and repackaging-based approaches at scale
- Assess third-party SDK risk within Android applications through automated code identification, vulnerability chain mapping, and contextual scoring
- Construct vector databases with multi-strategy query systems and auto-chunking for offensive security knowledge management
- Analyse patch diffs to identify incomplete fixes and generate payloads that circumvent new security checks
- Orchestrate multi-device test environments for automated, parallelised exploit verification

- Write effective prompts and build agentic workflows that reliably produce actionable offensive output — distinguishing where AI adds genuine value from where it introduces noise
- Extend the frameworks and methodologies taught in the course to new vulnerability classes, platforms, and research targets independently

Course Outline

Part 0 - AI Foundation [Labs: 1; Hrs: 1]

- Introduction to AI
- Commonly used AI models
- Model Context Protocol (MCP) servers
- Retrieval Augmented Generation (RAG)
- Setting up Claude Code - the right way (Lab)

Part 1 - Prompt Engineering [Labs: 1; Hrs: 1]

- Introduction to prompt engineering
- Building efficient prompts
- Reverse engineering apps with prompting (Lab)

Part 2 - Mobile Component Analysis [Labs: 1; Hrs: 2.5]

- Setting up the environment
- Detecting advanced vulnerabilities and risks in common attack surfaces
    - Exported components
    - WebViews
    - Deeplinks
    - Content Providers
    - Account Manager
    - Intents
    - Device logs
- Assessing advanced edge-cases using EdgeOfTheWorld app (Lab)

Part 3 - The AI Exploit Pipeline [Labs: 4; Hrs: 2]

- Automated payload generation
- Automated verification of exploitation
- Bypassing common defense-in-depth mechanisms
    - Root detection
        - Root detection bypass for common detection mechanisms
        - Complete root detection bypass using Frida (Lab)
        - Complete root detection bypass using repackaging (Lab)
    - SSL pinning
        - SSL unpinning for common pinning mechanisms
        - Complete SSL pinning bypass using Frida (Lab)
        - Complete SSL pinning bypass using repackaging (Lab)

Part 4 - SDK & Supply Chain Security [Labs: 1; Hrs: 1]

- Analysis of 3rd party SDKs

- Intelligent detection of 3rd party code within apps
- Contextual assessment of vulnerability chains from 3rd party to app
- Confused deputy exploitation
- “Gone-Rogue SDK” risk scoring
- Assessment of an open source SDK (Lab)

Part 5 - Advanced Agentic Workflows [labs: 5; Hrs: 4]

- Deterministic filtering + LLM decision making
- Multi-agentic architecture
- Bypassing advanced detections mechanisms
    - Building a VectorDB
        - Embedding models
        - Multi-strategy query systems
        - Auto-chunking
    - Radare2/GhidraMCP server for native detection layer
        - Detection of native C++ detection code
        - Bypassing native detection with Frida Interceptor
    - Building high quality and effective prompt (Lab)
    - Advanced repackaging at scale (Java + JNI) (Lab)
    - Case study: root detection bypass of top 100 global apps
- Self-learning loop: Delta detection and auto-updating data (Lab)
- Self-healing loop: Contextual multi-layer analysis, detecting failures and reconstructing payloads (Lab)
- Device Orchestration (Lab)
    - Device reboot, shutdown
    - Automated app lifecycle handling
        - Installing/uninstalling apps
        - Launching apps/app components
        - Log monitoring
        - Input simulation
- Advanced DAST
    - Cross-app communication monitoring
    - AI-based summarisation of app flows
    - AI-driven vulnerability detection

Part 6 - Vulnerability Research [Labs: 2; Hrs: 4]

- Analysis of system apps
    - Setting up the environment
        - Installing Android beta images
        - Android SDK setup for PoC generation
    - Case studies
        - Improper access control
            - CVE-2022-23998: Unauthorized camera access on a locked device
            - CVE-2022-28789: Unprotected activities leading to unauthorised audio recording

            - CVE-2022-30717: Improper caller check allows untrusted applications to use camera functions
            - Unintended PII leakage via Google App via exported content provider in Android 16
            - CRLF parameter smuggling allows arbitrary internal deeplink launch
        - Privilege escalation
            - SVE-2020-18025: Unauthorized access to files stored in secure folder
            - Private Space bypass in Android 16 allows enumeration of installed apps and trigger state changes for app permissions
        - Side channel data leakage
            - CVE-2022-22291: Allows attackers to access cell location via device logs
- Manual CVE Deconstruction (Lab)
    - Root cause analysis of a recent Android CVE
    - Mapping CVE advisories to decompiled source code
    - Patch analysis
- Reproduction of a CVE exploit (Lab)
    - Generating exploit for CVE-2023-35674: BAL bypass via VirtualDisplay Presentation window (**7.8 HIGH**)

Part 7 - Automated Exploit generation for N-Days [Labs: 2; Hrs: 3]

- Analyzing Android CVEs with AI (Lab)
    - Contextual mapping of CVEs to specific Android versions
    - Automated ingestion of security bulletins and advisory data
- Reproduction of CVE exploits (Lab)
    - AI-driven detailed analysis of the vulnerabilities
    - Automated PoC APK generation
- Automated verification of exploitation
    - Headless execution of PoC on target devices
    - Log analysis and detection for successful exploitation

Part 8 - Automated Pipeline for Android exploit generation [Labs: 1; Hrs: 4.5]

- Orchestrating the exploit lifecycle
    - Integrating MCP servers for real-time tool interaction
    - Building a deterministic feedback loop for failed exploit attempts
- Patch analysis of recent CVEs
    - Automated diffing of AOSP commits (Pre-patch vs. Post-patch)
    - Identifying "Incomplete Fixes" via LLM code analysis
- Patch bypass & automated verification (Lab)
    - Generating variant payloads to circumvent new security checks
    - Automated regression testing against patched builds
- Case study: CVE-2025-22429: BaseBundle Use-after-Recycle via lazy value count loss (**8.1 HIGH**)
- Case study: CVE-2025-48572: Background Activity Launch via Media Button Receiver (**7.8 HIGH**)

- Case study: CVE-2025-48596: Parcel OOB Heap Read (Phantom Memory) (**7.8 HIGH**)
- Case study: CVE-2025-48627: BAL via startNextMatchingActivity (**7.8 HIGH**)
- Case study: CVE-2023-20906: SYSTEM_ALERT_WINDOW Permission Retention (**7.8 HIGH**)
- Case study: CVE-2023-35674: BAL bypass via VirtualDisplay Presentation Window (**7.8 HIGH**)
- Case study: CVE-2023-20911: BaseBundle Parcel mDataSize Inflation via appendFrom (**7.8 HIGH**)

Part 9 - Analysing known CVEs for 0-days [Labs: 1; Hrs: 3.5]

- From N-Day to 0-Day: Variant Analysis
    - Using known CVEs as seeds for sink-and-source discovery
    - Automated scanning for similar patterns in 1st party system apps
- Advanced Case Studies (Lab)
    - Discovery of a 0-day variant from a 2025 CVE
    - Automated extraction and analysis of privileged system components
- Developing a self-learning Loop
    - Delta detection and auto-updating the VectorDB
    - Automated reporting and documentation generation

Topic Tags

Android security, Agentic AI, Offensive Security

Difficulty Level

Intermediate to Expert

Suggested Prerequisites

This course is designed to welcome participants with **diverse skill levels** - no formal prerequisites are required. However, the following foundational knowledge will help you get the most out of the hands-on labs

- A **basic understanding of Linux** (navigation, permissions, running commands)
- Basic understanding of **Mobile** ecosystem (apps, devices, emulators, OS)
- Fundamentals of **Android** applications (**Java**, **Kotlin**) and **Android Studio** IDE
- Familiarity with LLMs such as **Claude, Gemini** etc
- The ability to **run simple Python scripts** from the command line
- Claude Code Max or Codex or GLM subscription - preferably the $200 plan with cyber verification program completed for claude/codex

What the Trainer Will Provide

- Lab Virtual machines (VMs) with all required tools, dependencies, and datasets pre-installed
- Digital copies of all training material including slides & labs

- Source code access for all custom apps which was built for the training
- Access to private Slack channel for ongoing discussion, future support, and networking with instructors and fellow participants during/after the course