SYNTAX COLLEGE

4-DAY HANDS-ON COURSE · ATTACK & DEFENSE

Android Security

Analyze, attack and protect Android apps – from the OS security model and APK reverse engineering with JADX and Ghidra to malware analysis and traffic interception with Burp Suite.

DURATION	FORMAT	LEVEL	PREREQUISITE
4 days	**On-site, theory + labs**	**Developers & security pros**	**Basic Linux shell**

01 About the course

A four-day intensive course for experienced Android developers, reverse engineers and cybersecurity professionals who want to understand how Android systems and apps are analyzed, attacked and secured.

The course moves from the operating system's layered security model – kernel, sandbox, permissions and signatures – through hands-on analysis with Android Studio and ADB, to a live malicious-app demonstration, APK and native reverse engineering with JADX, Ghidra and apktool, and man-in-the-middle traffic interception with Burp Suite and certificate pinning.

Who should attend

- Security researchers and reverse engineers at cyber, mobile-security and forensics companies
- Android developers building payment, banking, enterprise or SDK products that must resist tampering
- Product-security and pen-testing teams

What participants will be able to do

- Explain Android's device, kernel and app security layers
- Use ADB to inspect devices, apps and abnormal behavior
- Find vulnerabilities in manifests, components and deep links
- Reverse-engineer APKs (Kotlin and native code) and spot malware
- Reverse-engineer native code with Ghidra and decode hidden secrets
- Intercept HTTPS traffic and harden apps against MITM

02 Day-by-day syllabus

DAY 1 The Android OS and its security model

- AOSP and Google Mobile Services
- Platform architecture: kernel, HAL, ART, NDK, framework, system apps
- JVM, DVM and ART
- Boot process with focus on Zygote
- Processes, threads and memory management
- File system overview
- Device, kernel and app security
- System permissions (signature | privileged)
- Platforms: Wear OS, Android TV, Android Automotive
- Security enhancements across Android releases

HANDS-ON LABS

- **System services from the shell** (~45m · hands-on · adb)
  List system services and invoke a Binder method directly with service call; inspect memory, permissions and SELinux state with dumpsys.
- **UIDs and SELinux live** (~30m · hands-on)
  Read your security context with id and watch SELinux denials appear in logcat while triggering a denied access.

DAY 2 Android Studio, developer signatures and ADB

Part 1 – Android Studio and signatures

- Project structure, build configurations and Gradle
- Profiling and the APK Analyzer
- APK signing with private/public keys
- Detecting a developer signature and finding apps that share it

Part 2 – ADB in depth

- Connecting to devices; installing and locating APKs
- File transfer; shell commands (am, broadcasts, services)
- Querying the device: RAM, installed apps and more
- Detecting abnormal app behavior and excessive resource use
- Key events; extracting files from an APK

HANDS-ON LABS

- **ADB drill** (~2h · hands-on · 3 exercises)
  Open the camera, maps and dialer through intents; a RAM walkthrough with a foreground service, dumpsys and force-stop; capture a photo through an intent and pull it from the device.
- **Developer signature analysis** (~45m · hands-on · scripts)
  Group every installed app by signing certificate – find debug-signed apps and a vendor platform certificate shared by hundreds of apps.
- **Normal vs abnormal app** (~45m · investigation)
  Detect an app that over-uses RAM, CPU and sensors with top and dumpsys, and an app targeting an outdated SDK.

DAY 3 Application security and a live malicious app

Part 1 – Components and the manifest

- App components, app states, intents and intent filters
- Manifest investigation with live demonstrations
- Permissions; deep links and how to protect them
- Location and Play services; foreground and bound services
- Storage, FileProvider and multithreading
- Summary: security, processes and IPC

Part 2 – Malicious app demonstration

- Speech recognition (ASR) as a data-collection vector
- Tricking the user into granting dangerous permissions
- A hidden background service with constant server communication
- Surviving reboots and hiding through system events
- Cloud storage of collected data
- Revealing the app from its manifest – and could this happen on iOS?

HANDS-ON LABS

- **Deep links** (~45m · hands-on)
  Open apps through deep links from adb, understand implicit intents and browser fallback, and see how unprotected links are abused.
- **Location & permissions** (~45m · hands-on)
  Grant and revoke location permission from adb and follow what the app can see.
- **Storage forensics** (~45m · hands-on)
  Pull files from shared, private-external and internal storage – including run-as on debuggable apps.
- **SMS receiver & service call** (~45m · hands-on)
  Trigger an SMS broadcast receiver and call system services directly from the terminal.
- **The hidden recognition app – capstone** (~2h · guided walk-through)
  A purpose-built spy app: speech recognition in the background, a boot receiver, screen and volume events, a foreground service – and how to reveal it from its manifest.

DAY 4 Vulnerabilities, reverse engineering and traffic interception

Part 1 – Vulnerabilities and malware detection

- Backdoors and dynamic code loading
- Hard-coded values and malware indicators
- APK reverse engineering – Kotlin and native C++ code (JADX, Ghidra, apktool)
- Encoding and encryption: Base64, Hex, XOR, RC4, AES
- Obfuscation with D8 and ProGuard/R8

Part 2 – Man-in-the-middle

- Burp Suite proxy setup on Android
- Installing a CA certificate with ADB
- HTTPS interception and capturing API traffic
- Network Security Config
- Certificate pinning – and how it stops the attack

HANDS-ON LABS

- **Backdoor footprint** (~30m · investigation)
  Find a backdoor's process, package and open sockets with dumpsys, ps and ss.
- **Dynamic code loading** (~45m · hands-on · d8)
  Convert a JAR to DEX and load code at runtime – and learn to spot it in a suspicious app.
- **Hard-coded native secrets** (~1h · hands-on · JADX + Ghidra)
  Find hard-coded keys across the Kotlin and native .so layers and identify an RC4-style stream cipher.
- **ProGuard / R8 obfuscation** (~30m · hands-on · JADX)
  Decompile minified vs unminified builds and see what obfuscation really hides.
- **Man-in-the-middle drill** (~2h · hands-on · Burp Suite, 13 phases)
  Proxy the phone through Burp, install the CA, capture and decode an Authorization header, forge and replay it in Repeater – then see certificate pinning stop the attack.

> **Bonus drill:** Optional module: dynamic instrumentation with Frida – runtime hooking, root-detection and certificate-pinning bypass.

18	4	20+	5	1	13
labs & drills	days	lab apps & APKs	pro tools	spy-app capstone	MITM drill phases

What participants receive

- Step-by-step drill pages with full solutions
- Purpose-built lab apps and APKs (spy app, backdoor, vulnerable and pinned apps)
- Presentations for every session
- Tools: JADX, Ghidra, apktool, Burp Suite, CyberChef (+ Frida in the optional module)
- Windows and macOS command coverage
- Android terms sheet and review questionnaire

03 Format & logistics

Sessions	4 full days
Hours	09:00–16:00
Prerequisite	Familiarity with Linux/Bash
Dates	Coordinated with your team

Requirements

- A training room with a projector or large screen
- A laptop per participant with Android Studio installed (setup guide sent in advance)
- Android devices or emulators; USB debugging enabled
- Up to 15 participants per group
- Test devices or emulators that may be rooted; Burp Suite (Community is sufficient)

All offensive techniques are taught for defensive, research and testing purposes on lab devices and purpose-built sample apps.

04 Your instructor

All Syntax College courses are designed and taught personally by **Eran Katsav**, founder of Syntax College. Eran holds a B.Sc. in Computer Science from the Hebrew University of Jerusalem and an M.Sc. with honors from Tel Aviv University, and has more than 20 years of experience training engineers in industry, the defense sector and academia.

He lectures on native Android development in Kotlin and Java at Reichman University and at the Holon Institute of Technology (HIT), where he also heads the mobile lab. As an independent developer he created **Bip it**, the first Hebrew voice assistant, which reached about one million downloads in Israel; white-label versions of its engine ship in all Suzuki vehicles sold in Israel.

Selected clients: **Samsung, General Motors, Cisco, ironSource, CWT, Ophir Optronics**. Authorized supplier of the Israeli Ministry of Defense.

Bring this course to your team
Tailored syllabus, on-site at your offices, in Hebrew or English.

Eran Katsav
+972-50-6944922
info@syntax.org.il
www.syntax.org.il