APPS BY VIKAS

Meet Vikas



GenzLudo

Ludo with 3D characters and up to 8 players. A first-of-its-kind no-teaming-up feature keeps play fair. Try it - you'll love the difference.

Download on Google Play



Kachu

Too much scrolling straining your neck? Add your addictive apps and complete daily neck exercises to unlock them.

Download on Google Play

01 / PROTECTION ROADMAP

Application Security & Anti-Reverse Engineering

Complete Roadmap to Protect Your App's Source Code, Logic, Assets, and Brand Name from Copycats — What to Do, Why It Matters, and AI Prompts.

By **Vikas Yadav** · VikasBuilds (vikasbuilds.com)

> **The Golden Rule:** You cannot legally protect a raw app idea. But you can 100% protect your **Source Code, Core Business Logic, Proprietary Assets, and Brand Identity**. Relying on a single tool (like a basic obfuscator) is never enough. Real security requires applying **6 Defense Layers** together.

Security Layer	What It Protects	Primary Defense
1. Backend Isolation	Core business logic, algorithms, secret API keys	Keep sensitive logic on secure server APIs
2. Build Obfuscation	Class names, functions, code readability	R8 / ProGuard, Flutter `--obfuscate`, symbol stripping
3. Anti-Tamper & RASP	APK repackaging, app cloning, Frida hooking	SHA-256 certificate check, Play Integrity API, freeRASP
4. Anti-LLM Trap	AI-assisted decompilation (Claude, Ghidra-MCP, JADX-AI)	Semantic stripping, opaque predicates, decoy functions
5. Network & Assets	API endpoints, media files, tokens, game assets	SSL Pinning, HMAC request signing, AES asset encryption
6. Legal & Brand IP	App name, logo, original UI & illustrations	Trademark registration (Class 9 / 41), copyright archive, DMCA

02 / LAYER 1

Backend Architecture & Secret Isolation

WHAT TO DO

- **Move Secrets to Server:** Move all private API keys, payment verification, subscription validation, reward/credit calculations, and proprietary algorithms from the mobile client to your backend server.
- **Client App as Display Layer:** Restrict your client app (APK/IPA) to rendering UI, handling smooth animations, and displaying verified data returned by your secure backend.

WHY IT MATTERS

Any code running directly on a user's mobile device can be examined through a debugger, runtime hooking, or a memory dump by an experienced reverse engineer. If your core business logic lives on a secure server and every critical action requires server-side validation, a copycat cannot replicate your product even if they decompile your entire APK.

PROMPT 1 / CLAUDE & CHATGPT

```
Act as a Senior Application Security Architect. I am sharing my client-side application code below. 1. Identify every hardcoded API key, secret token, pricing/reward calculation, and sensitive business logic that is currently exposed on the client side. 2. Explain how an attacker could exploit or clone each exposed piece. 3. Refactor the architecture by designing secure REST/GraphQL backend endpoints (Node.js/Python) to handle all sensitive logic server-side, and provide the updated client code that safely consumes these APIs using short-lived user session tokens. Here is my current code: [PASTE YOUR CODE OR FOLDER STRUCTURE HERE]
```

03 / LAYER 2

Build Obfuscation & Symbol Stripping

WHAT TO DO

- **Play Store Release Format:** Always upload `.aab` (Android App Bundle) instead of raw `.apk` and enable Google Play App Signing.
- **Native Android (Java/Kotlin):** Enable R8/ProGuard in release builds (`minifyEnabled true` and `shrinkResources true`) to eliminate unused code and rename classes and methods to short symbols (`a.b.c`).
- **Flutter Apps:** Build release bundles with obfuscation flags: `flutter build appbundle --obfuscate --split-debug-info=./debug-info`.
- **Web / JavaScript:** Apply minification, mangling, and control-flow obfuscation in production builds. Never upload source maps (`.map` files) to public web servers.

WHY IT MATTERS

Without build obfuscation, anyone can open your APK in beginner decompilers like JADX or Blutter and read your exact folder structure, function names (such as `unlockPremium()`), and business logic. Obfuscation scrambling makes static analysis tedious, confusing, and costly for attackers.

PROMPT 2 / CLAUDE & CHATGPT

```
Configure full production code obfuscation and symbol stripping for my [Flutter / Android Kotlin / React Native / Web] project. 1. Provide the exact build configuration files (e.g., build.gradle, proguard-rules.pro, or webpack/vite config) with minifyEnabled, resource shrinking, and log stripping (removing all console.log / Log.d calls in release). 2. Write custom ProGuard keep-rules so third-party SDKs (Firebase, Retrofit, Gson, etc.) do not crash while my core business classes remain heavily obfuscated. 3. Give me the exact CLI release build commands and explain how to securely store debug symbol files for crash reporting without exposing them in the final binary.
```

04 / LAYER 3

Anti-Cloning, Signature Lock & RASP

WHAT TO DO

- **Runtime Signature Verification:** At app startup, verify that the current SHA-256 signing certificate matches your official release key.
- **Installer Source Verification:** Verify the installer package name (`com.android.vending`) to detect unauthorized sideloading or unofficial third-party stores.
- **RASP & Play Integrity:** Integrate Google Play Integrity API and the freeRASP SDK to detect rooted devices, emulators, debuggers, and dynamic instrumentation tools (Frida / Xposed) and terminate suspicious sessions immediately.

WHY IT MATTERS

The most common mobile app theft is APK repackaging—where copycats decompile your APK, replace your AdMob publisher ID with theirs, and re-upload the clone. Because modifying the APK breaks the original cryptographic signature, enforcing runtime certificate verification causes cloned apps to shut down on launch.

PROMPT 3 / CLAUDE & CHATGPT

```
Write production-ready runtime tamper-detection and anti-cloning security code for my [Android Kotlin / Flutter] app. Include the following checks on app startup: 1. Verify the app's runtime signing certificate SHA-256 hash against my expected release certificate hash (store the expected hash obfuscated/split rather than a single plain string). 2. Verify the installer package name to detect unauthorized sideloading outside Google Play Store. 3. Detect if a debugger is attached, if the device is rooted/emulator, or if dynamic instrumentation tools like Frida/Xposed are active in memory. 4. If any check fails, gracefully block core API calls and terminate the session without showing obvious error messages that help attackers locate the check.
```

05 / LAYER 4

Anti-LLM Reverse Engineering Defense

WHAT TO DO

- **Opaque Predicates & Decoys:** Add opaque predicates (mathematical conditions that always evaluate to true/false but look complex in static analysis) and decoy functions around critical algorithms.
- **Strip Semantic Traces:** Remove all internal comments, debug strings, log statements, and descriptive error messages from production release binaries.
- **Layered Protection:** Always apply this as an additional trap on top of Layer 2 (Obfuscation), never as a standalone defense.

WHY IT MATTERS

Modern reverse engineers feed decompiled source directly into LLMs (Claude, ChatGPT, Ghidra-MCP, JADX-AI) to reconstruct architecture and logic in seconds. LLMs rely on semantic clues, method names, and linear control flow. Introducing dead branches and decoy logic confuses AI analyzers, resulting in incorrect summaries and broken clones.

PROMPT 4 / CLAUDE & CHATGPT

```
Harden the following critical module against automated AI/LLM-assisted reverse engineering (such as JADX-AI or Ghidra-MCP) while keeping runtime performance fast. 1. Strip all semantic hints, descriptive identifiers, and internal comments. 2. Introduce lightweight opaque predicates and non-trivial control-flow indirection around key validation logic so static LLM analyzers cannot easily summarize the function's intent. 3. Add 3 realistic-looking decoy functions that mimic business logic without ever executing in the main path. Ensure the output remains 100% bug-free and functionally identical to my original code: [PASTE CRITICAL MODULE]
```

06 / LAYER 5

API Traffic Lock (SSL Pinning) & Asset Encryption

WHAT TO DO

- **Certificate & Public Key Pinning:** Pin your backend server's public key hash in the mobile app client so proxy tools (Charles, Proxyman, mitmproxy) cannot inspect API traffic.
- **HMAC Request Signing:** Send a timestamped HMAC signature with every API request to prevent third parties from copying and calling your endpoints from unauthorized clones.
- **Secure Storage & Asset Protection:** Store auth tokens in Android Keystore / iOS Keychain. Encrypt custom game art, 3D models, audio, and configuration JSON files using AES/XOR before bundling.

WHY IT MATTERS

When attackers cannot easily read obfuscated code, they intercept network traffic with a man-in-the-middle proxy or unzip your APK to steal proprietary graphics, audio, and level designs. SSL Pinning and asset encryption shut down both attack vectors.

PROMPT 5 / CLAUDE & CHATGPT

```
Implement network and asset protection for my [Flutter / Android / iOS] application: 1. Write an HTTP client
setup (OkHttp / Dio) with Certificate/Public-Key SSL Pinning configured for my domain [yourdomain.com], including
a backup pin so certificate rotation does not break the app. 2. Provide a lightweight AES-256
encryption/decryption utility script (Python) to encrypt my bundled JSON/media assets before building, along with
the client-side runtime decryption loader. 3. Show how to store sensitive auth tokens using Android Keystore /
iOS Keychain instead of plain SharedPreferences/UserDefaults.
```

07 / LAYER 6

Name, Logo & Design Legal Protection (Brand Shield)

WHAT TO DO

- **Trademark Registration:** Register your app name and logo under Class 9 (Mobile Software & Apps) and Class 41 (Entertainment / Gaming) in your primary target jurisdictions. Display ™ upon filing and ® upon grant.
- **Timestamped Copyright Archive:** Maintain timestamped archives of Figma designs, original vector art, audio files, and private Git repository commit histories.
- **Domain & Social Handles Lock:** Secure your brand's official domain (.com, .io, .app) and major social handles prior to public launch.

WHY IT MATTERS

If your app gains traction without trademark protection, bad actors can register your brand name and demand your app's removal from app stores. With registered trademarks and timestamped source proofs, you can file official DMCA and IP infringement takedown notices that remove copycats within 48 to 72 hours.

PROMPT 6 / CLAUDE & CHATGPT

```
Act as an Intellectual Property & App Store Compliance Specialist. 1. Create a pre-filing Trademark checklist for registering my mobile app name "[APP NAME]" under Class 9 and Class 41, including how to verify conflicting marks on official trademark databases. 2. Draft a formal, legally crisp DMCA Copyright & Trademark Infringement Takedown Notice template that I can submit to Google Play Store, Apple App Store, or GitHub if a copycat clones my app name, icon, UI assets, or source code.
```

08 / PRE-LAUNCH CHECKLIST

Security Execution Checklist

Follow this sequence before releasing your app to production to ensure all security layers are active:

Order	Action Item	AI Prompt	Priority
Step 1	Move all API keys and core business logic to backend server	Prompt 1	Critical
Step 2	Enable R8/ProGuard or Flutter `--obfuscate` + log stripping	Prompt 2	Critical
Step 3	Enforce runtime SHA-256 signature verification & RASP integrity	Prompt 3	Critical
Step 4	Configure SSL Pinning and encrypt bundled assets with AES	Prompt 5	High
Step 5	Add semantic stripping and anti-LLM decoys to sensitive modules	Prompt 4	Recommended
Step 6	File trademark application (Class 9/41) and archive source proofs	Prompt 6	High

Built by VikasBuilds · vikasbuilds.com · @vikasbuildsapp
Need custom mobile app development, high-converting websites, Google Play Store publishing, or app security hardening? Visit **vikasbuilds.com** or connect on Instagram at **@vikasbuildsapp**.

More things worth exploring
Find more free guides, checklists, and developer tools at vikasbuilds.com/resources