Project Proposal: Android APK Static Security & Privacy Analyzer Web Application

# 1. Project Overview

This project proposes the development of a web-based application for static security and privacy analysis of Android APK files. The platform targets developers, security researchers, and educators by providing an automated tool to inspect APKs for insecure coding patterns, risky permissions, embedded third-party SDKs, and configuration issues. Additionally, the application integrates with VirusTotal to check the reputation of analyzed files using their SHA-256 hash.  
  
Users will upload APK files via a simple web interface and receive a detailed, downloadable report highlighting potential security and privacy risks.

# 2. Objectives

- Provide a secure, user-friendly web platform for analyzing Android APKs.  
- Perform static analysis to detect:  
 - Debuggable flags, hardcoded secrets, and use of HTTP  
 - Over-permissioning and sensitive hardware access (e.g., GPS, camera, microphone)  
 - Presence of risky or unverified third-party SDKs  
- Extract and inspect AndroidManifest.xml, certificates, and metadata  
- Integrate VirusTotal for threat intelligence via hash-based lookups  
- Rate applications based on a configurable risk scoring system  
- Generate user-friendly PDF reports summarizing findings

# 3. Methodology

The application will be built using Python with the Flask web framework. Users will upload APK files through a web interface. The backend will use tools such as zipfile, regex, apktool, and jadx to unpack and analyze APK contents. Static analysis will focus on metadata, manifest files, certificates, and embedded resources.  
  
The application will:  
- Extract APK metadata (permissions, components, intent filters, etc.)  
- Identify dangerous configurations using a local rule database (JSON)  
- Check for common privacy and security issues (e.g., trackers, insecure settings)  
- Compute a SHA-256 hash and query the VirusTotal Public API for detection status  
- Use WeasyPrint to generate PDF reports with visual summaries of results

# 4. Key Features

- File Upload Interface:  
 - Web-based interface with drag-and-drop and manual upload options  
 - Accepts `.apk` files only  
- Static Analysis Engine:  
 - Parse AndroidManifest.xml, resources, certificates, and smali code  
 - Use apktool and jadx to extract content  
 - Detect insecure configurations (e.g., android:debuggable, cleartext traffic)  
- Privacy Analyzer:  
 - Identify use of sensitive permissions and hardware access (e.g., location, camera)  
 - Highlight over-permissioning relative to app functionality  
 - Detect embedded third-party SDKs with known tracking behavior  
- VirusTotal Integration:  
 - Compute SHA-256 hash of the uploaded file  
 - Retrieve scan results via the VirusTotal Public API  
 - Display antivirus detection stats if available  
- Risk Scoring System:  
 - Assign overall risk levels (Safe / Moderate / High)  
 - Score breakdown by category (permissions, configuration, SDKs, etc.)  
- PDF Report Generation:  
 - Generate a detailed, well-formatted report using WeasyPrint  
 - Option to anonymize file names and hashes  
 - Enable download or short-term viewing

# 5. Tools and Technologies

- Backend: Python 3.x, Flask  
- Static Analysis: zipfile, hashlib, regex, apktool, aapt, jadx  
- PDF Generation: WeasyPrint  
- Threat Intelligence: VirusTotal Public API  
- Frontend: HTML/CSS (optionally extendable with React or Tailwind later)  
- Storage (optional): Temporary local/S3-compatible storage, PostgreSQL for results metadata

# 6. Feasibility and Considerations

- Feasibility: Static analysis of APKs is fully feasible using established tools like apktool and jadx.  
- Challenges:  
 - Ensuring safe handling and deletion of uploaded APKs  
 - Rate limits and API quotas for VirusTotal  
 - Keeping rule and SDK pattern databases up to date

# 7. Development Timeline (Prototype Phase ~2 Weeks)

- Week 1:  
 - Implement file upload system in Flask  
 - Integrate APK unpacking and parsing logic  
 - Create pattern matching engine and rule database  
- Week 2:  
 - Build frontend to display analysis results  
 - Integrate VirusTotal API  
 - Implement PDF report export with WeasyPrint  
 - Final testing and deploy MVP

# 8. Future Work

- Support dynamic analysis using emulators  
- Build SDK reputation database for more detailed tracker detection  
- Add REST API for CI/CD integration  
- Extend risk model with machine learning  
- Create admin dashboard for analysis trends and insights

# 9. Use Cases

- Security Researchers auditing APKs for vulnerabilities  
- Developers testing builds before publishing on Google Play  
- Educators teaching mobile security and privacy best practices  
- QA Teams validating apps for compliance and data handling standards