**Security Report: Evaluation of MobSF (Mobile Security Framework)**

**Subject:** Suitability of MobSF for organizational use in mobile application security assurance  
**Assessment target:** Wave2 Android application (APK)

## 1) Introduction

This assessment evaluates the Mobile Security Framework (**MobSF**) as a candidate tool for the organization’s mobile application security program. MobSF is an open-source platform for **static** and **dynamic** analysis of Android and iOS apps. Given the threat landscape for mobile (e.g., activity hijacking, insecure backups, excess permissions), the objective was to determine whether MobSF can reliably surface security defects and how it should be integrated into our SDLC and CI/CD processes.

## Executive summary

**Scan outcome for Wave2 (Android):**

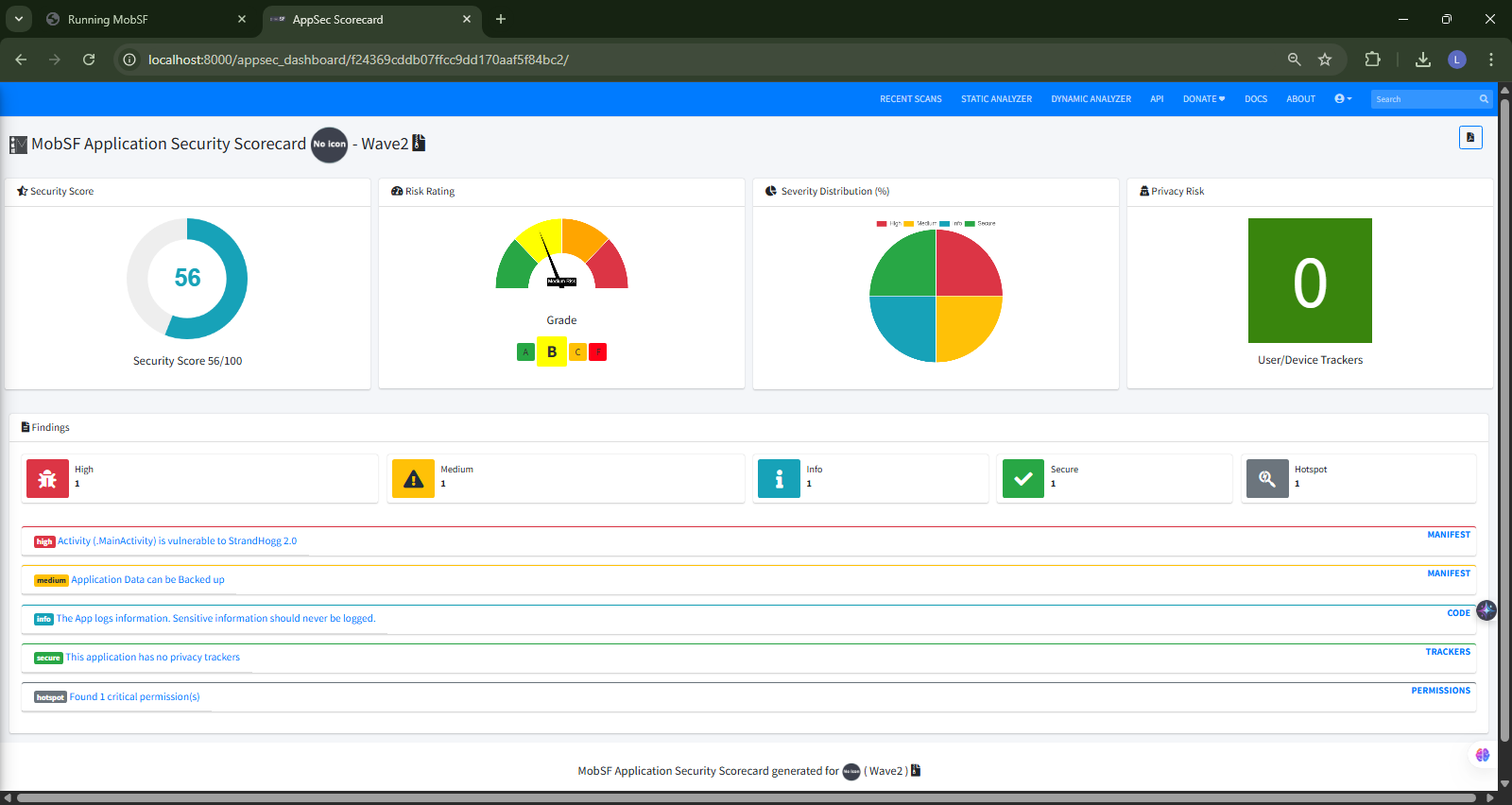
* **Overall security score:** **56/100**
* **Risk rating:** **Medium (B)**
* **Key issues:**
  + **High:** Indicators of **Strand Hogg 2.0** risk in MainActivity
  + **Medium:** **Application data backups enabled**
  + **Informational:** **Sensitive logging** detected
  + **Hotspot:** Presence of a **dangerous permission** requiring review
* **Positive:** No privacy trackers detected

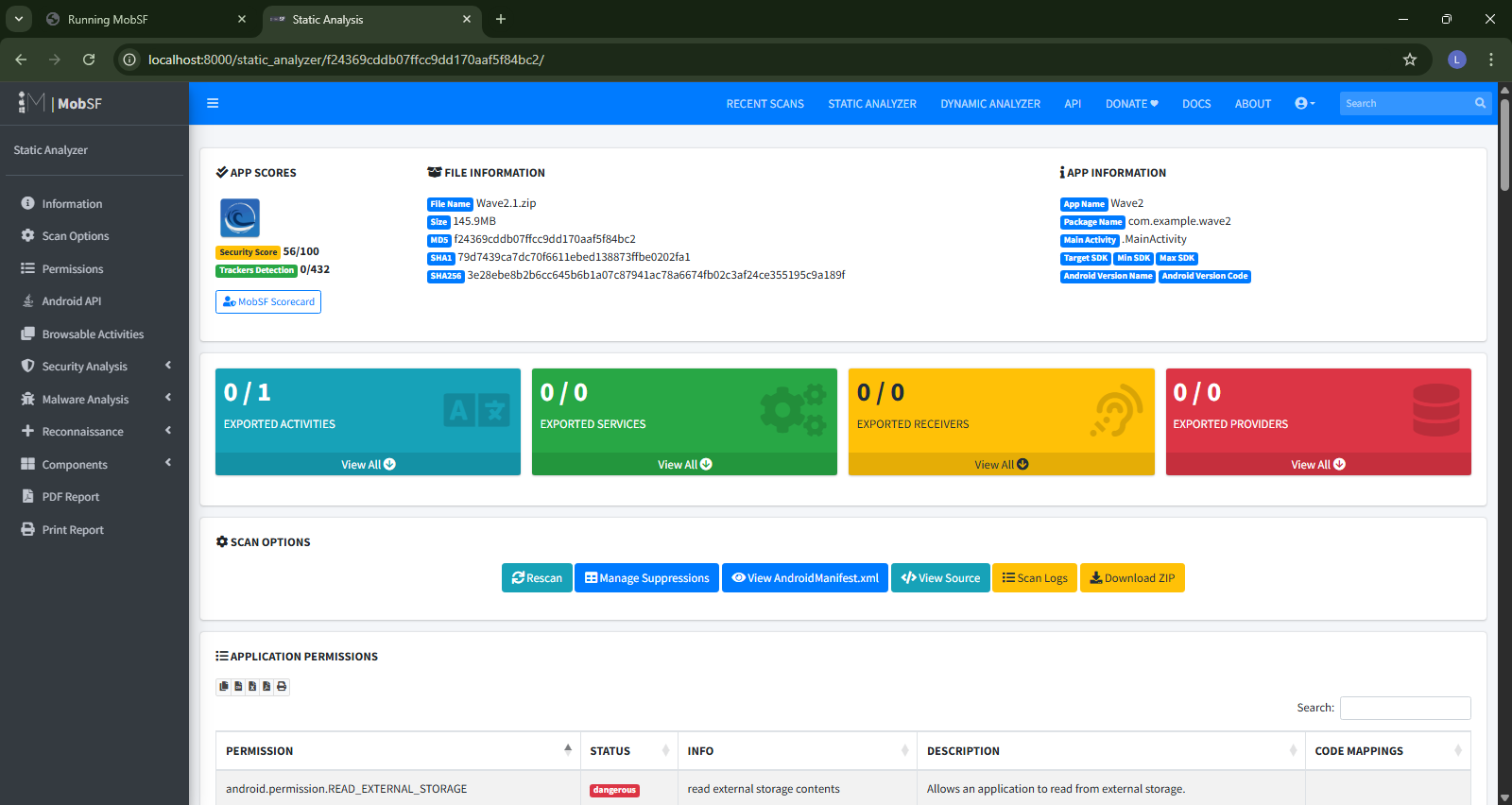
## Findings Snapshot

|  |  |
| --- | --- |
| Category | Result |
| Overall Score | 56/100 |
| Risk Rating | B (Medium Risk) |
| High Severity | 1 (StrandHogg 2.0) |
| Medium Severity | 1 (Backup enabled) |
| Info | 1 (Sensitive logging) |
| Secure | 1 (No trackers) |
| Hotspot | 1 (Critical permission) |

Reference: Full scan scorecard and details are attached in the original MobSF scorecard (Wave2).

Source (uploaded scan): MobSF\_Security\_Scorecard\_Report\_Wave2.pdf





## 3) Detailed findings

### 3.1 StrandHogg 2.0 indicators — **High**

* **Description:** Attackers can launch a malicious activity that impersonates a legitimate one, capturing user input.
* **Impact:** Credential theft and session compromise.
* **Evidence:** MobSF manifest/intent analysis flagged exported activity conditions consistent with the attack pattern.
* **Required remediation:** Review AndroidManifest.xml; set android:exported="false" for activities not intended for inter-app access; tighten intent filters and task affinities; add explicit authentication gates before sensitive screens.

### 3.2 Application data backup enabled — **Medium**

* **Description:** Default backup behavior can copy app data to cloud backups.
* **Impact:** Leakage of credentials/configuration if a user’s backup store is compromised.
* **Required remediation:** Set android:allowBackup="false" for sensitive apps, or implement encrypted/whitelisted backups with fullBackupContent.

### 3.3 Sensitive information logged — **Informational**

* **Description:** Diagnostic logging appears to include sensitive values.
* **Impact:** Possible disclosure via device logs or aggregated telemetry.
* **Required remediation:** Remove or mask secrets, tokens, and user identifiers; centralize logging with secure sinks and retention policies.

### 3.4 Dangerous permission present — **Hotspot**

* **Description:** The app requests at least one permission classified as dangerous.
* **Impact:** Broader attack surface and potential abuse if not strictly needed.
* **Required remediation:** Conduct a permission audit, follow least-privilege, and justify each dangerous permission in the design record.

## Prioritized Action Plan (30/60/90 day)

|  |  |  |
| --- | --- | --- |
| Timeline | Priority | Tasks |
| 0–30 days | Critical | Patch StrandHogg activities; re-scan before release |
| 30–60 days | High | Disable backups; review logs; reduce permissions |
| 60–90 days | Medium | Integrate MobSF into CI/CD; scheduled scans; developer training |

## 4) Tool evaluation

### Strengths

* Clear, actionable reports with risk scoring suitable for non-security stakeholders.
* Strong **Android-specific** checks (e.g., StrandHogg patterns, manifest hardening).
* Simple **web UI** for ad-hoc review and a **REST API** for automation.
* Export to **PDF/JSON** supports audit evidence and change-control packs.

### Limitations / Considerations

* Heavily **static-analysis** focused; some issues only surface at runtime.
* Findings may include **false positives**; requires triage and developer context.
* Effective use presumes basic mobile security knowledge and secure-coding standards.

**Conclusion on the tool:** MobSF is suitable as a **first-line automated control**. It should complement, not replace, manual secure-code review and penetration testing.

## 5) Deployment & usage guidance

### 5.1 Recommended deployment (Docker)

docker pull opensecurity/mobile-security-framework-mobsf:latest

docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest

 Access the console at http://localhost:8000.

 Upload APK ⇒ **Static Analyzer** ⇒ Review findings ⇒ Export report.

### 5.2 Source install (alternative)

* Clone repository → create Python virtualenv → pip install -r requirements.txt → start server → access web UI.

Automation via API (CI/CD)

# Upload

curl -F "file=@app-release.apk" \

-H "Authorization: MobSF <API\_KEY>" \

http://localhost:8000/api/v1/upload

# Trigger scan

curl -X POST -d "file=<uploaded\_file>" \

-H "Authorization: MobSF <API\_KEY>" \

<http://localhost:8000/api/v1/scan>

## 6) Remediation & adoption roadmap

**Days 0–30**

* Fix **StrandHogg** indicators (exported activities, intents, task affinity).
* Re-scan and attach MobSF report to the change-request (CR).

**Days 30–60**

* Disable or encrypt **backups**; remove **sensitive logs**; complete **permission audit**.
* Introduce secure-logging configuration and redaction guidelines.

**Days 60–90**

* Integrate **MobSF API** into Android CI pipeline with quality gates.
* Deliver **secure-coding training** focused on Android manifest hardening and permission minimization.
* Define acceptance criteria: **0 High**, **0 Critical**, and **no new Medium** findings prior to release.

## 7) Recommendation

Adopt **MobSF** as an approved security scanning tool for mobile applications and integrate it into the CI/CD pipeline with enforceable gates. Maintain **manual penetration testing** and **threat modeling** for release candidates handling sensitive data. Proceed with remediation of the Wave2 findings and schedule a follow-up assessment after fixes are merged.

**Reference List**

OpenAI. 2025. ChatGPT.

[Large Language Model].

Available at:  
<https://chatgpt.com/share/68c7efa6-085c-8012-9d5f-cf25bcfe1efd>  
[Accessed 13 September 2025].

Mobile Security Framework (MobSF).

[Online]. Available at: <https://mobsf.github.io/docs/#/>

[Accessed 13 September 2025].

### Annexure

**Title:** Disclosure of AI Usage in My Assessment  
**Section within the assessment in which it was used:** MobSF Security Assessment Report – Wave2 Android Application  
**Name of AI tool:** ChatGPT  
**Purpose of use:** To generate a structured report based on the MobSF test findings, summarizing vulnerabilities, risk ratings, remediation plans, and recommendations  
**Date of use:** 13 September 2025  
**Link:** <https://chatgpt.com/share/68c7efa6-085c-8012-9d5f-cf25bcfe1efd>