Project Report: Android Sandbox Security & Forensic Analysis System

Team Members:

- Deekshith
- Skanda
- Pratham
- Aftab

---

1. Executive Summary

The **Android Sandbox Security System** is a cutting-edge mobile application designed to empower security operations center (SOC) analysts and forensic investigators with a robust tool for analyzing suspicious Android files and behaviors directly on-device. In an era where mobile threats are becoming increasingly sophisticated, the need for a portable, secure, and detailed analysis environment is paramount. This project bridges the gap between complex desktop-based forensic tools and the accessibility of mobile platforms.

Currently, the project has established a high-fidelity frontend interface built with **React Native**, serving as the control center for the analysis capabilities. The application simulates the workflow of submitting files (such as APKs), analyzing their metadata, and presenting actionable security intelligence in a clear, SOC-grade dashboard. Crucially, the app includes a native Android integration layer to perform real-time permission analysis and risk scoring of installed applications.

The completed vision of this application includes a fully integrated backend capable of **Tor traffic correlation**, **PCAP (Packet Capture) analysis**, and **AI-driven malware detection**. By combining static analysis with network behavioral monitoring, the Android Sandbox aims to provide a comprehensive security assessment for high-risk environments.

---

2. Introduction

2.1 Background

Mobile devices are now the primary computing platform for billions of users, making them a prime target for malicious actors. Traditional antivirus solutions often rely on signature-based detection, which fails against novel or polymorphic malware. Furthermore, investigating the network behavior of these applications—specifically traffic routed through anonymizing networks like Tor—requires specialized setups that are typically hard to deploy.

2.2 Project Objective

The primary objective of this project is to develop a user-friendly yet technically advanced Android application that enables:

1. **Static Analysis**: Rapid assessment of file properties, hashes, and permissions.
2. **Behavioral Monitoring**: Tracking application activity and network connections.
3. **Forensic Reporting**: Generating detailed, legally defensible reports for security incidents.
4. **Network Attribution**: Correlating traffic flows to identify malicious command and control (C2)

servers, even when obscured by Tor.

2.3 Scope

The current phase of development focuses on the **User Experience (UX)**, **Interface Logic**, and **Native Bindings**. We have successfully built the application framework, navigation flow, and the visualization engine for security reports. We have also implemented the Java-based native modules (BehaviorMonitor, NetworkAnalyzer, MalwareDetector) that bridge the gap between the React Native UI and low-level Android system APIs.

---

3. Literature Survey

In the domain of mobile security, several tools exist, but most are designed for desktop environments or lack specific focuses on anonymized traffic.

Tool	Platform	Primary Focus	Limitation
MobSF (Mobile Security Framework)	Web/Desktop	Static & Dynamic Analysis	Requires heavy server setup; not portable on a phone.
Cuckoo Sandbox	Server	Malware Sandboxing	Complex deployment; infrastructure heavy.
Virginis	Android	General Antivirus	Relies heavily on signatures; lacks forensic network analysis.
Android Sandbox (Our Solution)	**Android (Native)**	**Forensics & Tor/Network Correlation**	**Portable, SOC-focused UI, specifically targets traffic attribution.**

Our solution uniquely targets the niche of *on-device forensic triage* combined with advanced network correlation concepts, which is not typically found in consumer antivirus apps.

---

4. Methodology & Technical Implementation

The system employs a specific methodology to analyze applications, broken down into Static Analysis (Permissions) and Behavioral Analysis.

4.1 Hybrid Architecture (React Native Bridge)

The core innovation in our implementation is the **React Native Bridge**. While React Native handles the UI connected to the user, it cannot access low-level Android system details directly. We implemented a Custom Native Module (BehaviorModule.kt) to solve this.

How it works:

1. **JS Realm**: The UI requests a scan (e.g., BehaviorModule.analyzeApp('com.example.malware')).
2. **The Bridge**: React Native serializes this request and passes it to the Java Native Interface (JNI).
3. **Native Realm**: The BehaviorModule class invokes helper classes:
    - BehaviorMonitor: Scans permissions.
    - NetworkAnalyzer: Checks for cleartext traffic and data exfiltration risks.
    - MalwareDetector: Runs heuristic checks.
4. **Response**: The data is packed into a WritableMap and sent back to the JS realm to be visualized.

4.2 Permission-Based Heuristic Analysis

The BehaviorMonitor class implements a weighted risk scoring algorithm. Instead of just listing permissions, we categorize them based on potential for abuse.

Algorithm:

1. **Classification**: Permissions are divided into HIGH_RISK (e.g., Camera, Mic, Contacts, SMS) and MEDIUM_RISK (e.g., Storage, Location).
2. **Scoring**:
    - High Risk Permission = **15 points**
    - Medium Risk Permission = **5 points**
3. **Aggregation**: Total Score = $\min(100, \sum (High \times 15) + (Medium \times 5))$
4. **Verdict**:
    - Score $\ge$ 50: **HIGH (BLOCKED)**
    - Score $\ge$ 25: **MEDIUM (RESTRICTED)**
    - Score {{content}}lt;$ 25: **LOW (ALLOWED)**

This heuristic approach allows the app to flag zero-day malware that abuses permissions, even if its signature is not yet known.

4.3 Behavioral Network Analysis

The NetworkAnalyzer module (integrated into BehaviorModule) assesses the app's network posture:

- **Cleartext Traffic**: Checks if the app allows HTTP (non-HTTPS) traffic, which is a security vulnerability.
- **Data Exfiltration**: Checks for combinations of READ_CONTACTS + INTERNET, creating a high "Data Exfiltration Risk Score".

---

5. System Design

The application follows a **Client-Server-Controller** architecture designed to separate the UI from the heavy computational logic.

5.1 Architecture Diagram

```
graph TD
    User[User / Analyst] -->|Interacts| UI[React Native UI]
    UI -->|JS Bridge Call| Native[Create Native Module]

    subgraph "Android Native Layer (Java/Kotlin)"
        Native -->|Invoke| Monitor[BehaviorMonitor]
        Native -->|Invoke| Net[NetworkAnalyzer]
        Native -->|Invoke| Malware[MalwareDetector]

        Monitor -->|PackageManager| System[Android System API]
        Net -->|Permission Checks| System
    end

    subgraph "Backend (Planned)"
        Traffic[Tor Traffic Logs] -->|Ingest| Analysis[Python Analysis Engine]
        Analysis -->|Correlate| DB[Threat Database]
    end

    Native -->|Return Data| UI
```

5.2 Data Flow Diagram (Scan Process)

```
sequenceDiagram
    participant User
    participant App as React Native App
    participant Bridge as Java Bridge (BehaviorModule)
    participant Monitor as BehaviorMonitor

    User->>App: Clicks "Scan App"
    App->>App: Show Loading State
    App->>Bridge: analyzeApp("com.target.app")
    Bridge->>Monitor: getAppPermissions()
    Monitor-->>Bridge: List[Permissions]
    Bridge->>Bridge: Calculate Risk Score
    Bridge-->>App: { risk: "HIGH", score: 85, action: "BLOCK" }
    App->>User: Display SOC Report
```

---

6. Current Functionality (The Prototype)

We have completed the **Frontend Interface** and the **Native Analysis Layer**.

6.1 Home Dashboard

The landing screen provides immediate situational awareness.

- **Quick Scan**: A prominent "Shield" button allows users to instantly initiate a scan.
- **Recent Activity**: Displays a list of recently analyzed files with efficient caching.

6.2 Scan Result Interface (SOC-Style)

This is the core feature implemented. Upon "scanning" a file, the app presents a detailed report inspired by professional SOC (Security Operations Center) tools.

- **Risk Scoring**: Implemented a RiskBadge component that categorizes threats.
- **Confidence Metrics**: A dynamic progress bar showing the AI/Heuristic verdict confidence.
- **Actionable Intelligence**: The app suggests actions (ALLOWED, BLOCKED, RESTRICTED).

6.3 Implementation Details

- **File**: screens/ScanResultScreen.js
- **Native Module**: com.tempandroidsandbox.BehaviorModule
- **Tech**: React Native 0.83, Kotlin 1.9

---

7. Intended Functionality (The Completed App)

The final version of the app will connect these UI elements to live security modules. The "Completed App" will perform the following sophisticated tasks:

7.1 Real-Time Malware Detection

The app will integrate true malware detection logic.

- **Signature Matching**: Comparing file hashes (SHA-256) against global threat intelligence feeds (VirusTotal, ThreatFox).
- **Heuristic Analysis**: Identifying suspicious code structures even in unknown (zero-day) files.

7.2 Tor Traffic Correlation & Forensics

A unique feature of this project is its ability to analyze anonymized traffic.

- **Dual-Side Correlation**: The system will correlate traffic entering the Tor network (Entry Nodes) with traffic exiting it, allowing the identification of hidden servers.
- **Flow Logs**: Conversion of raw PCAP data into flow logs for easier analysis.
- **Defensive Forensics**: Providing "legal-grade" evidence chains for law enforcement use cases.

7.3 AI-Driven Risk Assessment

- Replacing the current static logic with a **Siamese Neural Network**.
- This AI model will learn from traffic patterns to predict malicious intent with high accuracy, minimizing false positives.

---

8. User Interface & Experience (UI/UX) Design

A significant effort was placed on the visual identity of the application. Security tools are often clunky and difficult to use; we aimed for the opposite.

- **Dark Mode First**: The interface uses a deep, professional dark theme (#1a1a1a backgrounds) that reduces eye strain for analysts working long hours.
- **Color Psychology**:
    - **Red**: Reserved strictly for confirmed threats and blocks to demand attention.

    - **Green**: Used for safe, allowed actions to provide reassurance.
    - **Blue/Grey**: Used for neutral information and metadata.

- **Typography**: We utilized clean, sans-serif fonts to ensure readability of dense hash data and technical logs.

---

9. Testing & Validation

To ensure the reliability of a security tool, we implemented a rigorous testing strategy.

9.1 Unit Testing

- **JavaScript**: We use **Jest** to test React components, ensuring that risk badges render the correct color for a given input string.
- **Kotlin**: JUnit tests verify that the BehaviorMonitor correctly calculates scores (e.g., verifying that a CAMERA permission triggers the correct score increase).

9.2 Integration Testing

- **Bridge Testing**: We verified that data types sent from Java (Maps, Arrays) are correctly deserialized into JavaScript Objects.
- **Error Handling**: Confirmed that the app gracefully handles cases where an analyzed app is uninstalled or has no permissions, returning a "Unknown" or "Low Risk" state without crashing.

9.3 Manual Verification

- **Test Case**: We installed a known "Benign" app (Calculator) and a "High Risk" test app (permissions for Mic, Camera, Contacts).
- **Result**: The Scanner correctly flagged the High Risk app with a Red badge and the Calculator with a Green badge, validating our heuristic algorithm.

---

10. Conclusion

The **Android Sandbox Security System** is well on its way to becoming a premier tool for mobile forensic analysis. We have successfully built a robust, extensible frontend that visualizes complex security data in an intuitive manner, backed by a native Android analysis layer.

The foundation laid by the team—**Deekshith, Skanda, Pratham, and Aftab**—is solid. With the UI/UX finalized and the Native Bridge operational, the next stages will focus on wiring this beautiful interface to the powerful backend analysis engines. Once completed, this tool will not only detect malware but provide deep insights into the network behavior of adversaries, offering a capability rarely seen in mobile applications.

The project demonstrates a high level of technical competence, combining modern mobile development (React Native) with advanced cybersecurity concepts (Traffic Correlation, Sandboxing). It stands as a testament to the team's ability to tackle full-stack security engineering challenges.