International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

Android Apps Intrusion Inspection

Prof. Patange S. P.[1], Sanket Yadav [2]

[1]*Professor, Computer Science Engineering, SSIET Ghogaon, Maharashtra, India*
[2]*Student, Computer Science Engineering, SSIET Ghogaon, Maharashtra, India*

Abstract - *Smartphones have become central to modern life, enabling communication, financial transactions, education, entertainment, and productivity. However, the increasing use of mobile applications introduces significant security and privacy risks. Many users unknowingly install applications that request excessive permissions, access sensitive personal data, or execute background tasks without consent. Traditional mobile security solutions, such as antivirus software, require continuous scanning, frequent updates, and high computational resources, making them unsuitable for lightweight devices. This paper presents Android based application developed in Kotlin that detects and notifies users of suspicious applications installed on their devices. The system leverages Android's Package Manager API to retrieve installed applications and compares them with a predefined list of flagged package names. If a match is detected. the user receives an immediate high-priority notification. The application also provides a user-friendly Interface, a splash screen, and a settings module for configuration. Its rule-based detection system ensures fast and reliable identification of known suspicious applications. The proposed approach is efficient requires minimal system resources, and enhances user awareness by providing real-time security alerts without requiring root access or complex background processing.*

Key Words: Android Security, Android Apps innovation, Suspicious Android Inspection, Data Suspicious Inspection

1. INTRODUCTION

Smartphones are no longer just communication devices; they serve as platforms for managing personal finance, healthcare, education, and social interaction. Android, as an open-source operating system, dominates the global market due to its flexibility, affordability, and extensive application ecosystem. However, this openness has also introduced severe security challenges

Third-party applications, especially those iinstalled outside the official Play Store, may bypass standard security mechanisms. Even applications from the official store may request excessive permissions that compromise use privacy. Applications requesting access to sensitive data such as contacts, messages ccamera, microphone, location, and storage-car misuse these privileges for unauthorized data collection, monitoring, or performing tasks in the background without the user's knowledge

Traditional antivirus and security solutions depend heavily on signature-based scanning cloud updates, and continuous monitoring. While effective in detecting known threats, these approaches require significant system resources impacting battery life and device performance and are often unsuitable for lightweight or low end devices.

The increasing sophistication of malware iincluding zero-dav attacks, ransom ware, spyware, makes it imperative to develop lightweight, real-time detection systems. Mobile users need an efficient mechanism to inspection ppotentially harmful applications without relying on heavy processing or continuous backgroundd scanning.

The lightweight, rule-based Android application that monitors installed apps and alerts the user about suspicious activity. By leveraging the Package Manager API, the application retrieves a list of installed applications and compares them to a predefined database of suspicious packages. If a match is found, the system generates a high-priority notification, enabling users to take prompt action.

The application includes a splash screen to enhance user experience and a settings module for customization. Its design emphasizes simplicity, efficiency, and user empowerment. B, combining real-time notifications with a light weight detection mechanism.

2. RELATED WORK

Permission-Based Inspection

Permission-based analysis is one of the earliest and most widely studied techniques in Android malware detection. Android applications must request permissions in their manifest file to access sensitive resources such as contacts SMS, camera, microphone, and location Researchers hypothesize that malicious applications often request excessive or irrelevant permissions compared to their functionality

A foundational study in this area is Felt et al (2011), "Android Permissions Demystified." The authors analyzed how Android users understand permission requests and found that users frequently grant permissions without fully understanding their security applications. This weakens the effectiveness of permission-based security mechanisms.

Following this, multiple studies applied machine learning techniques such as Support Vector Machines (SVM), Naive Bayes, and Random Forest classifiers using permission sets as feature vectors. These models demonstrate moderate detection accuracy in controlled datasets.

- Permissions do not reflect actual runtime behavior of applications
- Legitimate apps may request sensitive permissions, causing false positives
- Malware can request permissions dynamically after installation
- Users often ignore or misunderstand permission warnings

Lightweight Intrusion Inspection

Lightweight IDS research focuses on building security systems that operate efficiently on resource-constrained mobile devices. These systems aim to balance detection capability with minimal performance overhead

- On-device processing instead of cloud dependency
- Rule-based classification
- Event-driven architecture
- Minimal background services

Examples from research include lightweight host- based IDS models that monitor system calls, CPU usage, and application events

- Lack of real-time user notification systems
- Limited usability for non-technical users
- Weak integration between detection logic and user interface
- Poor adoption in real-world mobile applications

The proposed Android Apps Intrusion Inspection addresses these limitations by:

- Operating without root access or heavy computation
- Implementing a whitelist-based trust model
- Providing immediate user notifications
- Maintaining a lightweight architecture suitable for low-end devices

Unlike traditional research systems that focus on deep analysis, It practical for everyday Android users.

3. COMPONENTS

The Android Apps Intrusion Inspection is built using a modular architecture in which each component performs a specific function to ensure efficient detection of suspicious applications on Android devices. The system components are designed to be lightweight, independent, and easy to maintain.

User Interface

The User Interface (UI) component is responsible for interaction between the system and the user It provides a simple and intuitive design that displays application data and detection results

Functions:

1. Displays splash screen during applicationn launch
2. Shows list of installed and detectedd applications
3. Provides settings screen for configurationn
4. Enhances user experience with clear navigation

This component ensures that users can easily understand detection results without technical complexity.

Detection Engine

The Detection Engine is the core logic unit of the system.
Functions:

1. Maintains a predefined suspicious application list
2. Compares installed applications with stored data
3. Uses linear matching algorithm for detection
4. Flags applications as "suspicious" when matched

This component ensures fast and efficient detection with minimal processing overhead

Notification

This component handles alert generation when a suspicious application is detected
Functions:

1. Uses Android Notification Manager API
2. Generates high-priority notifications
3. Displays application name and warning message
4. Alerts user in real time

It ensures immediate user awareness of potential threats.

Data Storage

The Data Storage component manages all local data used by the system

Functions:

1. Stores suspicious application list locally
2. Uses Shared Preferences for settings storage
3. Ensures data privacy within the device
4. No external server or cloud storage is required

Settings

The Settings component allows user-leve customization and control

Functions:

1. Enable or disable notifications
2. Refresh application scan
3. Manage detection preferences
4. Control system behavior

This component improves flexibility and user control.

4. INPUT-PROCESS-OUTPUT

The Input-Process-Output model is one of the most fundamental system design frameworks. In the context of mobile application development, particularly for security focused applications, the IPO model plays a crucial role in defining system behavior. internal data flow, and functional execution.

Input

The input stage is responsible for collecting all necessary data required for detection and analysis. Inputs are primarily gathered from the Android operating system using system-level APIs.
Installed Applications Data

1. This is the primary input source of the system It includes:
2. Application package names
3. Application labels System
4. Application metadata

This data is retrieved using the Android Package Manager API

Process

The process stage is the most critical part of the system. It defines how input data is transformed into meaningful output through a structured detection mechanism
The processing module performs

1. Data filtering
2. Data comparison
3. Pattern matching
4. Decision making Result generation

Data Collection

1. Installed applications are collected using Package Manager API

2. Data Extraction extracts:
Package name App labell
System flags

3. Comparison Operation
Each installed application is compared with the suspicious feature

4. Detection Decisionn
If match found - mark as suspicious If no match mark as safe

5. Result Storage
Detected applications are stored temporarily ir memory.

6. Output
Results are sent to notification and UI modules.

Output

The output stage represents the final result generated by the system after processing input data.

5. METHODOLOGY

The methodology of Android Apps Intrusion Inspection is based on a lightweight, rule-based detection approach for identifying potentially suspicious applications installed on an Android device. The system is implemented using Kotlin and utilizes Android's core APIs such as Package Manager for retrieving installed applications and Notification Manager for generating alerts. Provide fast detection, minima resource consumption, and real-time user notification without requiring continuous background execution.

Detection Process

The detection process is performed using a linear comparison algorithm, where each installed application's package name is checked against the suspicious application list. If a match occurs. the application is marked as "Issue App or "Suspicious App." This approach is simple efficient. and suitable for real-time execution or mobile devices

Notification

Once a suspicious application is detected. the system immediately triggers notification using the Notification

Manager API The notification contains the application name and a warning message, ensuring that the user is promptly informed about potential security risks

User Interface

The detected applications are displayed in a structured user interface using components such as Recycler View. This allows users to view flagged applications along with relevant details like application name and package identifier

Execution Flow

The overall workflow of the system is as follows

1. Application launch via splash screen
2. Initialization of scanning module
3. Retrieval of installed applications
4. Comparison with suspicious database
5. Identification of matched applications
6. Notification generationn
7. Display of results in the user interface

6. IMPLEMENTATION

The Android Apps Intrusion is implemented on the Android platform using Kotlin within Android Studio. The system is designed to be lightweight, responsive, and compatible with a wide range of Android devices.

The Implementation focuses on real-time detection of suspicious applications, efficient notifications and an intuitive user interface

1. Splash Screen:

1. Provides a visual loading screen when the application is launched
2. Enhances user experience and allows background initialization of scanning modules.

2. Application Scanner:

Uses the

1. Package Manager get Installed Applications() method to fetch all installed applications.
2. Collects app metadata such as package name, label, and system flags.
3. Filters out system applications if the scar focuses on user-installed apps

3. Detection Engine

1. Maintains a local database of suspicious application package names
2. Iterates over installed applications and compares package names against the database.
3. Flags any application found in the list as suspicious.

4. Notification:

Generates notifications using the Notification Manager API Notifications include:

1. Application name
2. Warning message Action button to open the app list
3. Ensures the user is immediately alerted about potential threats

5. User Interface:

1. Displays detected applications in a Recycler View.
2. Shows app name, icon, and status (suspicious/found).
3. Supports user interaction such as removing apps from the list or refreshing scans

6. Settings:

1. Allows users to enable/disable notifications, configure scan preferences, and refresh the database.
2. Stores settings locally using Shared Preferences

7. Adaptable

The app can work on different Android versions without major changes

1. New "issue app detection rules" can be added easily
2. The system can integrate future security databases
3. The Ul can be modified without changing the core logic Notification features can be enhanced later

8. CONCLUSION

Android Apps Intrusion provides a lightweight, efficient solution for detecting suspicious Android applications and notifying users in real time. It complements traditional antivirus systems by providing instant awareness on low-end devices. While the current system is rule-based, it establishes a foundation for future development incorporating machine learning, cloud intelligence, and behavioral monitoring.

REFERENCES

[1] S. Patel and R. Shah. "Rule-Based Intrusion Detection in Mobile Devices," International Journal of Computer Science Engineering, vol. 7, no. 4 2019.

[2] D. Singh and R. Yadav, "An Overview of Android Malware Detection Techniques," International Journal of Computer Applications vol. 182, no. 30, 2019.

[3] A. Gupta and N. Sharma, "Security Challenges in Mobile Operating Systems", International Journal of Advanced Computer Science, vol. 11. no. 2,2020.

[4] A. Singh and M. Verma, "Android Application Security and Permission Analysis," International Journal of Information Security Research, vol. 10 no. 4, pp. 201-210, 2020.

[5] S. Gupta and R. Mehta, "Lightweight Intrusion Detection Techniques for Mobile Devices." International Journal of Computer Applications vol. 176, no. 5, pp. 15-21,2018.

[6] P. Rao and K. Nair, "Behavior-Based Malware Detection in Android Systems," Journal of Cyber Security Studies, vol. 8, no. 2, pp. 99-110, 2022.