International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

WEAKNESS OF PERMISSION SYSTEM IN CLOUD CONNECTED MOBILE APPLICATION

[1]Dr. Chetan Shelke, 2Rathnakar Achary, 3Dr. Rahul D. Ghongade, 4Dr. Virendra Kumar Shrivastava,

[1]Associate Professor, Department of Computer Science and Engineering, Alliance School of Advanced Computing, Alliance University, Bangalore, India, 562106
chetan.shelke@alliance.edu.in

[2]Alliance School Of Advanced Computing, Alliance University, Bangalore Indian.
rathnakar.achary@alliance.edu.in

[3]Professor, Department of Electronics and Telecommunication Engineering, P. R. Pote Patil College of Engineering & Management, Amravati

[4]Professor, CoE in CV, Department of Computer Science and Engineering,

Alliance School of Advanced Computing, Alliance University, Bangalore, India, 562106

Abstract

When mobile applications are utilized in cloud-connected environments, they often seek an excessive amount of permissions, which may lead to security and privacy concerns. Malicious applications have the potential to misuse sensitive data that is accessible to over-privileged programs, hence revealing vulnerabilities in the authorization method. According to the findings of this study, a multi-tiered detection framework that employs machine learning techniques in combination with signature verification, user input, and permission-based analysis is recommended for the purpose of determining if a program is dangerous, non-malicious, or suspicious. In terms of detection accuracy, the results of the experiments demonstrate that the proposed system outperforms the approaches that are currently being used, while at the same time reducing the number of false positives. According to the findings of the research, a strategy that is capable of effectively securing mobile applications contained inside ecosystems that are connected to the cloud might be discovered by combining permission analysis with approaches that are behavioral and user-driven.

Keywords: Mobile Security, Cloud-Connected Applications, Permission-Based Analysis, Malware Detection, User Feedback, Machine Learning, Over-Privileged Apps, Signature-Based Detection.

1. Introduction

Most individuals now use Smartphone apps for communication, banking, and entertainment. The advent of cloud-connected mobile apps, which store and process user data remotely, has raised privacy and security concerns. Android and other similar systems' security relies on the permission system. It controls access to private data including contacts, whereabouts, messages, and device storage [1].

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1.1 Permission System in Mobile Applications Only those with user-granted privileges may access the application's resources. This safeguards critical data from unauthorized access. Developers will ask users to authorize rights in the program manifest file [2] during installation or runtime. This method purportedly prevents programs from accessing sensitive data without user consent. **1.2 Weaknesses in Permission Systems**

Even though it's crucial, the authorization system has a few flaws:

- **Over-privileged Applications:** The attack surface is increased since many applications ask for permissions that aren't necessary for their basic operation [3].
- **User Awareness and Consent Issues:** Malicious apps may easily take advantage of the system since users often provide rights without fully comprehending their consequences.
- **Dynamic Threats:** The inability to identify malicious behaviour at runtime or via cloud interactions is a result of the static nature of traditional permission models.
- **Permission Misuse:** Data collection, spamming, and remote assaults are all possible outcomes of abusing sensitive rights like ACCESS_FINE_LOCATION, READ_CONTACTS, and SEND_SMS [4].

1.3 Significance of the Study

To build more effective security frameworks, we must know where the authorisation system is weak. Permission use patterns, user input, and machine learning algorithms may be used to detect malicious or overprivileged programs before user data is compromised. This method is crucial since cloud-connected apps often transport and store sensitive data.

2. Objectives

1. To find and examine permission system flaws in mobile apps linked to the cloud, with an emphasis on overly privileged and possibly dangerous apps.
2. To create a multi-layered detection system that effectively identifies malware by integrating machine learning, user input, and authorization analysis.

3. Research Methodology

Several mobile malware detection concepts exist. Examples of similar concepts: **3.1 Permission-Based Analysis**

App permissions control Android system resources and user data. Apps cannot access sensitive data by default; users must provide rights during installation. While developers disclose needed permissions in AndroidManifest.xml, applications sometimes seek more rights than necessary, increasing security concerns and making malicious activity harder to detect. Studies suggest that over one-third of programs are overprivileged, requiring thorough permission examination.

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Several permission-based malware2 detection methods exist. Lightweight approaches 1determine malignancy scores from manifest file permissions, intent filters, and process metadata. Apps over a threshold are recognized as harmful with 90% accuracy. Permission-based malware detection compares requested and needed permissions and uses Naive Bayes, SVM, and Decision Trees on big datasets to identify over 81% of malware. Permission analysis tools like PUMA and Kirin have high detection rates but occasional erroneous positives.

Advanced approaches statically analyze Dalvik bytecode control and data flows to find malicious functions. SCANDAL, DroidMOSS, and AndroGuard use clustering or classification algorithms to identify malware and repackaged applications using API calls, intents, and permission use. ComDroid targets app communication vulnerabilities.

To find abnormalities, machine learning methods like K-means clustering split programs by permission consumption. Although successful, these approaches are computationally expensive, cannot identify dynamically loaded malware, and struggle with reflective or obfuscated code. Permission-based analysis is still essential for detecting vulnerabilities in cloud-connected mobile apps, protecting against malware and privacy breaches.

3.2 Feature Selection Methods Remove unusable features (attributes) to minimize dataset size. Selection of remaining features is based on dataset feature representation capacity. Efficient feature selection techniques improve classification performance, thus it's frequently a prerequisite for dataset preparation. Feature selection strategies reduce dataset size, effort, and analysis time. For this reason, feature selection is crucial to dataset preparation for analysis. Choosing a suitable feature selection approach is becoming increasingly crucial in dataset analysis, altering the study's analysis phase. Two attributes—Gain Ratio, ReliefF—and two subsets—Cfs, Consistency—are used. Datasets are prepared via subset feature selection. Five Bayesian, Decision Tree, Classification and Regression Tree, J48 Decision Tree, Random Forest, SVM, and Sequential Minimal Optimization (SMO) methods are also selected. **3.3 Detection Techniques**

3.3.1. Signature-Based Detection

Signature-based malware detection compares computer code to a database of dangerous code signatures. If it matches, the program quarantines or blocks the file. This method is extensively utilized in commercial antivirus solutions since it detects known viruses well. It cannot identify new varieties or code-obfuscated malware and needs regular signature database upgrades.

Several methods have been developed to enhance signature-based detection. AndroSimilar analyzes variable-length application signatures to a malware database to detect dangerous programs, although its tiny database may overlook undiscovered infections. DroidAnalytics generates method-, class-, and application-level signatures at the opcode level to identify malware and repackaged programs, although shared signatures with innocent apps

Received: August 09 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

may cause false positives. API call sequences and similarity functions let SAVE identify malware variants better than commercial antivirus programs.

The template-based technique by Christodorescu et al. represents malware as instruction-variable-constant tuples and uses control flow analysis to find variations with few false positives. SAFE creates harmful pattern automatons to accurately test executables.
Honeycomb generates network-based signatures using honeypots, whereas MEDiC disassembles programs to ASM code and compares key/value instruction pairs against malware signatures to identify obfuscation.

Finally, hybrid signature-based detection detects self-encrypting and polymorphic viruses using static and dynamic analysis. Decrypting the virus in an emulator and doing static analysis to identify harmful system calls detects malware that avoids signature-matching. Signature-based detection is still essential for detecting known malware, but its inability to handle novel, disguised, or polymorphic threats has led to hybrid and semantic-aware technologies.

3.3.2. Anomaly-Based Detection

Crow Droid: Ikeret al. [5] suggested dynamically detecting application activity. Stracetool collects app
system calls, and a crowdsourcing program on the device writes a log file and uploads it to a distant server. Log files may include device information, programs, and system calls. 2-mean clustering is used on the server to categorize applications as malware or benign. Results are in server database. The solution analyzes deeply and requires plenty of resources. The client app must be installed on the user's device, and normal apps may be classified as malware if they make excessive system calls. Andromly: Shabtaiet al. [6] offered behavior-based Android malware detection. While running, it
monitors device state features like battery level, CPU consumption, and more to classify the app as benign or malware and then uses machine learning algorithms to distinguish between the two. The solution can identify and inform users of ongoing assaults.

Zhao's SVM-based malware detection system, AntiMalDroid [7], can detect harmful applications and variations during execution. First, it watches application behavior and attributes, then classifies them as normal or malicious. It then loads the two kinds of attributes into learning module and creates behavior signatures. Then it compares the signature to malware and benign app signatures in the database. The app is classified as benign if its signature matches current benign app signatures. High detection rate and dynamic signature database extension are possible with the approach. However, detection takes longer.

Taint Analysis :

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Encket al. [8] suggested TaintDroid for Android system-wide information flow monitoring. It can detect numerous sensitive data sources including camera, GPS, and microphone and identify third-party developer app data leaks. If sensitive data leaves the device, it identifies it and alerts the app. It efficiently tracks sensitive data but not control flow. It cannot trace information leaving device and returning in network reply. ***3.3.3. Specification-Based Detection*** Uses a preset set of regular rules to determine whether a software is harmful. Therefore, programs that break the rules are harmful. Spec-based malware detection solves pattern- matching deficiencies using an algorithm. This virus detection technique uses instruction semantics. The method resists regular obfuscation [9-11]. Malware's malevolent actions are sequences of variables and symbolic constants described using a template. Program attributes cannot be adequately stated using this technique. Anomaly-based detection evolved into specification-based detection. Specification-based detection approximates application or system requirements, not implementation. Specification-based systems train to learn all valid behavior of a program or system to examine. Main problem of specification-based system is that it's hard to correctly characterize system or application behavior. Panorama collects the system-wide information flow of a program under examination and examines its behavior against a legitimate set of rules to identify malicious activities [12]. **3.4 Feature selection methods** It reduces dataset size by deleting unusable characteristics (attributes). Selection of remaining features is based on dataset feature representation capacity. Efficient feature selection techniques improve classification performance, thus it's frequently a prerequisite for dataset preparation. Feature selection strategies reduce dataset size, effort, and analysis time [13-14]. For this reason, feature selection is crucial to dataset preparation for analysis. Choosing a suitable feature selection approach is becoming increasingly crucial in dataset analysis, altering the study's analysis phase. Thus, datasets are prepared using two attribute-based (Gain Ratio, ReliefF) and two subset-based (Cfs, Consistency) feature selection approaches. Five Bayesian, Decision Tree, Classification and Regression Tree, J48 Decision Tree, Random Forest, SVM, and Sequential Minimal Optimization (SMO) methods are also selected. **3.5 Classification Analysis** Malware detection utilizing classification and clustering machine learning methods is common. Many mobile malware datasets are analyzed using decision trees, Bayesian classifiers, and support vector machines. Based on permissions, API calls, and behavior, classification algorithms classify programs as benign or malignant.

Bayesian Classification estimates class membership probability using the Naive Bayes method, which implies attribute independence. Fast and reliable identification of suspicious apps is possible with its processing efficiency and ability to filter big datasets [15].

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Based on splitting criteria like information gain, CART and J48 decision tree algorithms build trees. Using minimum cost-complexity pruning, CART creates smaller, interpretable trees and handles category and numerical data without linearity. J48, an ID3 extension, classifies datasets using entropy-based splitting.

Random Forests (RF) use bootstrap aggregating (bagging) to generate numerous decision trees and output the majority class. They effectively estimate missing data, handle huge datasets with many variables, and remain accurate with incomplete datasets. Permission-based malware detection benefits from RF's classification accuracy [16].

SMO trains SVMs. It efficiently handles huge datasets by breaking quadratic programming issues into smaller subproblems. Linear memory and speed make SMO suited for large-scale malware detection. It normalizes characteristics and effectively handles nominal data for reliable categorization.

The general algorithm processes are described as following;

( ) Σ ΣΣ ( )

Subject to:

$0 \leq \leq C$ for i = 1, 2, 3,…, n,

$\Sigma = 0$

Where C is an SVM hyperparameter and K(xi, yj) is a kernel function. The variables αi areLangrange multipliers.

The algorithm proceeds;

1. Find a Langrange multiplier α1 that violates the Karush-Kuhn-Tucker (KKT) conditions for the optimization problem.

2. Pick a second multiplier α2 and optimize the pair (α1, α2)

3. Repeat steps 1 and 2 until convergence.

Classification analysis utilizing machine learning may detect malware in cloud-connected mobile apps by identifying patterns in permission use and application behavior [17].

3.6 Architecture Design of the Proposed System

Application signatures are initially checked against a harmful signature database in the proposed system. If no match is detected, it uses K-means clustering to compare app permissions to malicious and benign patterns to calculate a match score for each device (IMEI). Positive and negative comments for installed programs are pre-processed and coupled with signature and authorization ratings to include user input [18–19]. A J48 decision tree classifies apps as harmful, non-malicious, or suspicious, and users are warned. Based on user input, this multi-layered technique detects known malware, over-privileged programs, and possibly hazardous apps.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

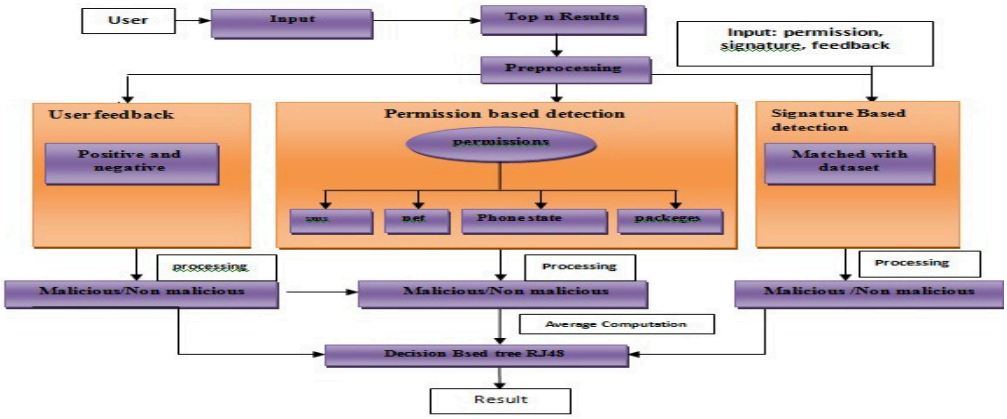


Figure 1. Architecture Design of the Proposed System

3.7 Permission-Based Detection Methodology

The proposed Android permission-based malware detection framework analyzes app permissions and classifies apps as harmful or benign to safeguard users. Android's Package Manager and Broadcast Receiver record newly installed or updated apps' permissions, reducing CPU and battery utilization. A server-side component compares extracted permissions to a permission database and groups dangerous and benign patterns using K- Means clustering. Key sensitive permissions like SEND_SMS, READ_PHONE, and INSTALL_PACKAGES are checked since they may access personal data or install backdoors [20, 21]. The system uses machine learning to categorize programs based on match ratios, user input, and other information to identify over privileged or possibly hazardous apps without running them.

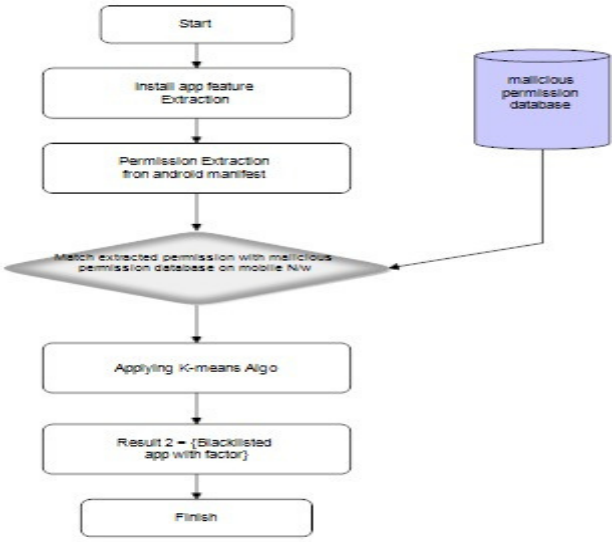


Figure 2. Permission based malware detection

3.8 Signature-Based Detection Methodology

The user-level component produces applicationsignatures using the MD5 technique, which compresses any amount of data into a 128-bit hash value indicating the app's digital

Received: August 09 2025 1334

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
fingerprint. They are subsequently forwarded to the mobile network component for comparison to a malicious app signature database. If a match is detected, the user is warned the app is harmful. The RIPPER method analyzes intents and opcode patterns to identify malware in new or unfamiliar programs without signatures in the database [22]. Using the database's continual maintenance, server-based signature detection provides centralized updates, accurate pattern matching, and smooth user alerting.

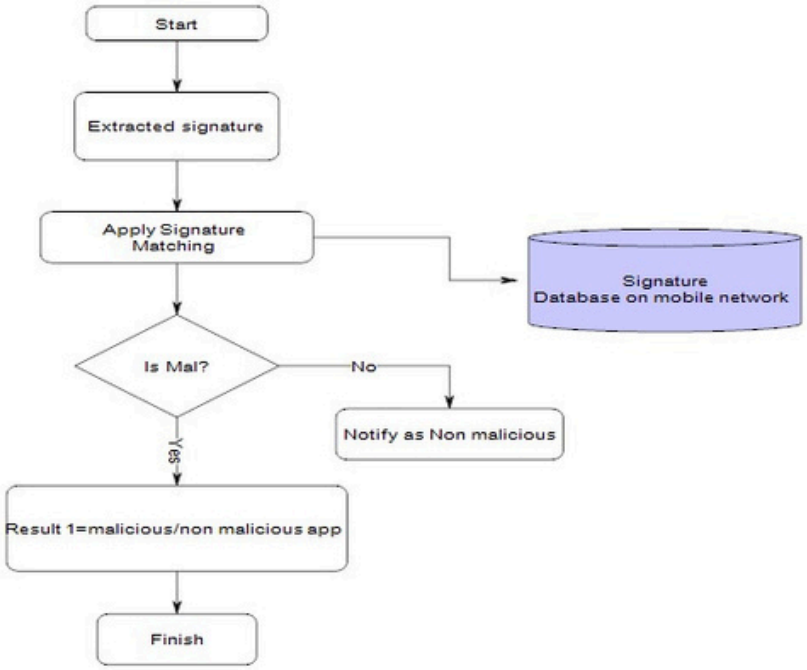


Figure 3: Signature Based malware detection

3.9 Performance Evaluation

Performance evaluation ofthe proposed approach is done based on classification of permission based and user feedback values

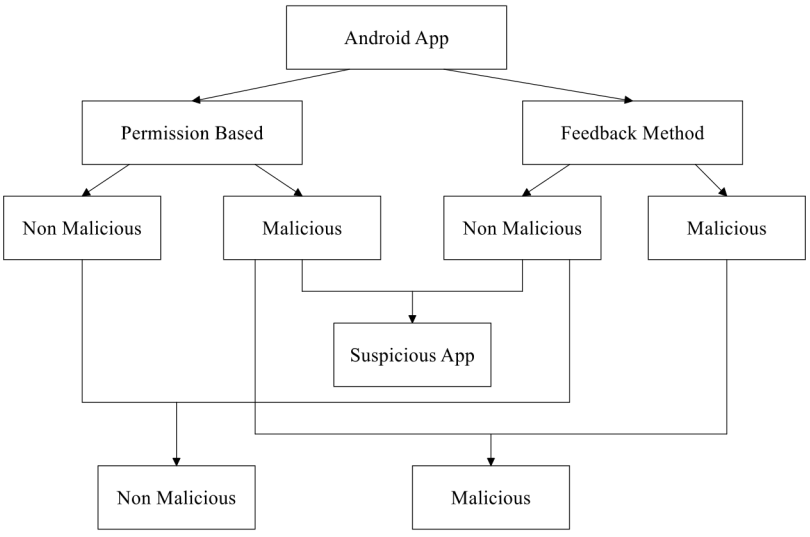


Figure 4 : Classification by RJ48 decision based tree

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Based on categorization context scenario, suggested technique performance is evaluated. J48 algorithm analyzes harmful, non-malicious, and suspicious applications. J48 is a basic C4.5 decision tree classifier. It makes a binary tree [23]. Classification problems benefit most from decision trees. This method models categorization using a tree. Each database tuple is classified using the tree once it is generated.

INPUT:

D //Training data

OUTPUT

T //Decision tree

DTBUILD (*D)

{

T=Android App;

T1= Create leaf node and label with permission based detection;

D= Database created by applying splitting predicate to malicious and non malicious;

T2= create leaf node and label with user feedback method;

J48 ignores missing values while generating a tree since its value may be anticipated from the record attribute values. The objective is to partition the app into harmful and non-malicious results depending on the item's training sample attribute values. J48 classifies using decision trees or their rules.

Malicious: The application is considered to be malicious if the both method i.e user feedback and permission based detection gives the result that app is malicious

Malicious +malicious=malicious

Non malicious: The application is considered to be an non malicious if both method i.e permission basec and user feedback method gives result as non malicious [24].

Non malicious +Non malicious=Non malicious

Suspicious: The application is considered to be suspicious if one of the method either permission based or user feedback method gives the result as malicious application.

Malicious + Non Malicious=Suspicious

4. RESULT 4.1 User-Level Module

4.1.1 Signature Extraction The signature-based detection method finds backdoors by creating each application's MD5 hash and string matching with harmful signature database hashes. Users extract application signatures per package using the MD5 technique, which compresses application data into a

Received: August 09 2025 1336

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
cryptographic hash [25]. These extracted signatures are submitted to the mobile network server over HTTP for examination and comparison to the malicious signature database to determine whether the application is hazardous

4.1.2 Permission Extraction

Permission-based scan on user level requires extracting all packages and permissions from mobile phone using the provided technique [26]. When the user selects a program from the list, its whole status and permissions are presented so the user can decide whether to retain or remove it. Users may access application statistics, and the system has automated permission- based malware detection that sends all extracted permissions over the mobile network and analyzes each program using K-means algorithm. Experimental investigation extracts 240 packets per IMEI and sends them to mobile network.

4.1.3 User Feedback Collection

User-level application feedback lets users score the app and provide good or negative comments. Methods of User Feedback: Android has many users that utilize and have troubles with apps. In the feedback mechanism, users may identify programs as harmful or non-malicious and provide feedback to mobile network servers to determine their score [27]. System marks a program as malware if the amount of negative feedback is over a specific threshold. System warns user of possible security risk if program is malicious. **4.2 Mobile Network Component**

The mobile network stores extracted signatures, permissions, and user input. Signature matches indicate a harmful app, while RIPPER and K-means algorithms discover apps with fewer false positives. Each input signature is systematically compared to all database signatures [28, 29]. ***4.2.1 Signature-Based Detection*** In signature based detection at mobile network malicious signature dataset is stored at mobile network extracted signature of app is then matched with malicious signature dataset. ***4.2.2 Permission-Based Detection***

At the mobile network, extracted permissions for each package are compared with stored malicious permissions, and a match classifies the app as malicious

Table 1: Extracted Permissions of Apps

Sig ID	Package Name	Selected Permissions
1079	APP - A	READ_CONTACTS, WRITE_CALL_LOG, SEND_SMS,

Received: August 09 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

		ACCESS_FINE_LOCATION
1045	APP - B	RECORD_AUDIO, INTERNET, WRITE_SMS, ACCESS_COARSE_LOCATION
869	APP - C	READ_CONTACTS, WRITE_CONTACTS, CAMERA, CALL PHONE, ACCESS_FINE_LOCATION
903	APP - D	READ_EXTERNAL_STORAGE, CAMERA, SEND_SMS, ACCESS_NETWORK_STATE
824	APP - E	RECEIVE_USER_MODE, INTERNET, WRITE_SETTINGS

The table lists important permissions for chosen Android apps. APP - A seeks critical rights like contacts, SMS, and GPS, raising privacy concerns. APP - D and APP – B need network, storage, and media access to operate. APP - C contacts, calls, and location might be misused. The table shows how permission analysis identifies malware-prone programs.

Table 2: List of Malicious Permissions in Database

Sr. No.	Permission Name	Malicious?
1	READ_EXTERNAL_STORAGE	Yes
2	READ_SMS	Yes
3	RECEIVE_SMS	Yes
4	SEND_SMS	Yes
5	WRITE_SMS	Yes
6	SYSTEM_ALERT_WINDOW	Yes
7	READ_PHONE_STATE	Yes
8	RECEIVE_BOOT_COMPLETED	Yes
9	INTERNET	Yes
10	CAMERA	Yes
11	CALL_PHONE	Yes
12	ACCESS_FINE_LOCATION	Yes
13	WRITE_CONTACTS	Yes
14	CHANGE_WIFI_STATE	Yes
15	WAKE_LOCK	Yes

Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
The table lists key permissions commonly exploited by malicious apps, such as SMS, camera, location, and network access. Any app requesting these permissions is flagged as potentially harmful for further analysis.

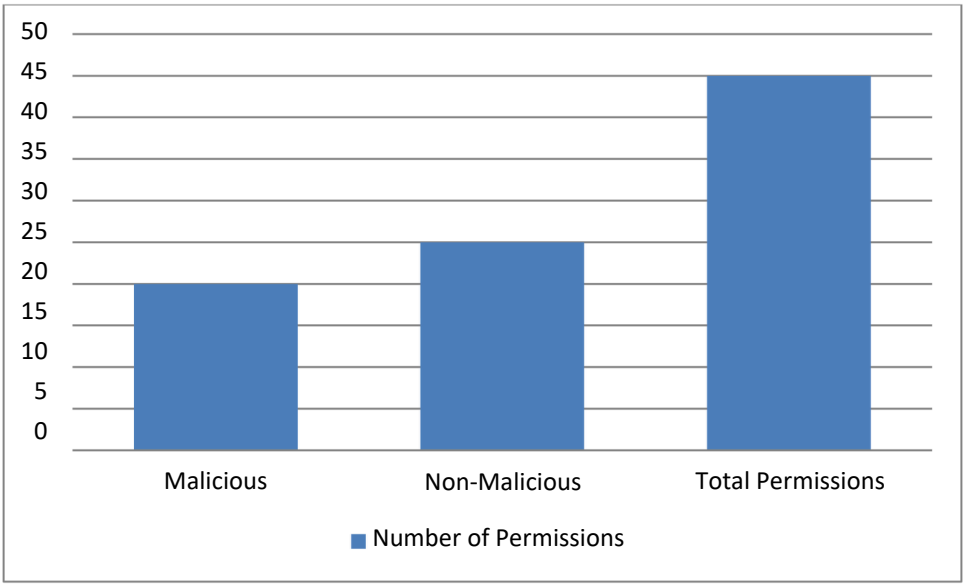


Figure 5: Analysis of Malicious, Non-Malicious, and Total Permissions

4.2.3 User Feedback Analysis

From each and every IMEI, number of users can give the positive or negative feedback about the applications which are stored on their mobile phone, these feedback are send over mobile network, on mobile network calculation of positive feedback ratio and negative feedback ratio is done [30-31].

Table 3: Positive and Negative Feedback from Users

AppName	Positive	Negative	IMEI
APP - D	10	1	8678020270269XY
APP - D	7 5	1	8634090307854XY
APP - A	7 0	1	8678020270269XYZ
APP - C	1 1	6	3568230731743XYZ
APP F	1 1	1	356823073174XYZ
APP - G	1	1	356823073174XYZ
APP –H		1	3568230731743XYZ
APP - I		0	000000000000000
APP - J		0	000000000000000
APP - K		0	000000000000000

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

APP - L	1	0	000000000000000
APP - M	0	2	000000000000000
APP - N	1	0	863409030785XYZ
APP - O	2	0	863409030785XYZ

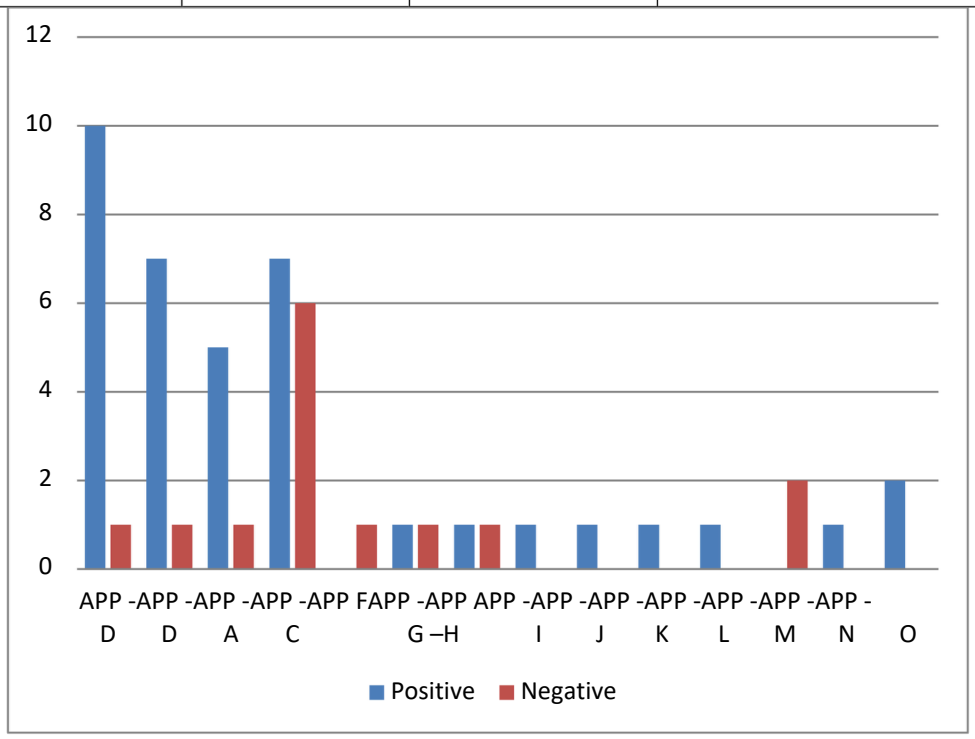


Figure 6: Number of Permissions Required per App

Table 4: Analysis of permission required by app

Application	Permission Required
APP - D	34 25 34 13 22 16
AAP - P	
APP - A	
APP - Q	
APP - L	
APP - R	

Table 4 shows that apps like APP - D and APP - A request the most permissions, indicating higher access to device resources and potential privacy risks, while APP – Q and APP - R require the least, reducing security concerns [32-35]. Overall, more permissions may offer richer functionality but also increase the app's vulnerability to misuse or malware

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

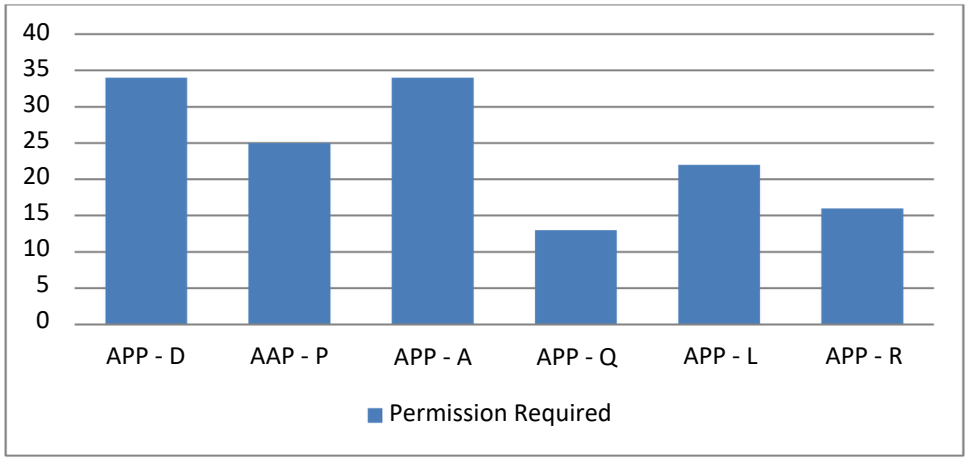


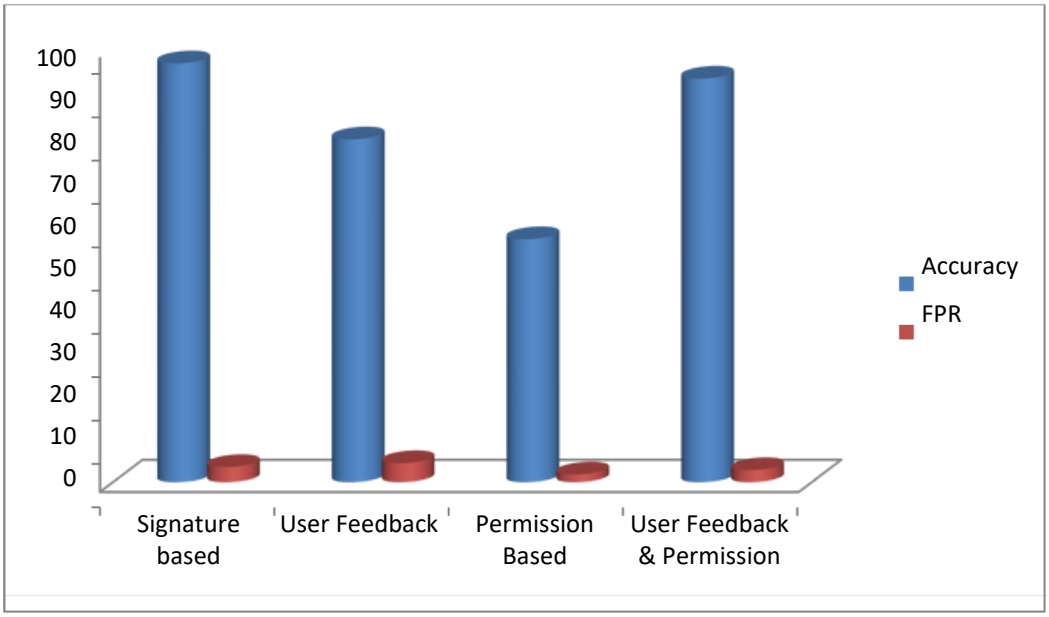
Figure 7: Analysis of permission required by app

4.3 Performance Evaluation

Table 5 : Accuracy for signature, user feedback, permission methods

Sr. No.	Method	Accuracy	FPR
1	Signature based Detection	96.5 %	3.5 %
2	User feedback method	79 %	21 %
3	Permission based detection	56 %	44 %
4	User feedback &Permission based detection	93 %	7 %

Table 5 shows that signature-based detection achieves the highest accuracy (96.5%) with the lowest false positive rate, while permission-based detection performs poorly (56% accuracy, 44% FPR); combining user feedback with permission analysis improves accuracy significantly to 93% with a low FPR [36-40].



Received: August 09 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
The table and graph above show that the proposed detection technique is more accurate than others. The proposed detection system has 0.8 accuracy, greater than others. After submitting 5 apps to various detection algorithms, 3 were malicious and 2 benign. The suggested approach detects 2 dangerous and 2 benign applications with 80% accuracy.

5. Conclusion

The research exposes the permission system shortcomings of cloud-connected mobile apps, which seek unnecessary permissions, raising security and privacy issues. Over-privileged programs may access SMS, contacts, location, and network resources, which malicious apps can use. The suggested system detects harmful, non-malicious, and suspicious programs using permission-based analysis, signature-based detection, and user input with machine learning. Experimental findings show that our multi-layered strategy increases detection accuracy, decreases false positives, and minimizes over-privileged app dangers. The study shows that behavioral analysis, crowd-sourced feedback, and classification algorithms improve mobile security over permissions alone. The research concludes that authorization system flaws must be addressed to secure cloud-connected mobile environments and user data against malicious exploitation.

6. References

1. Enck, W., Gilbert, P., Chun, B. G., Cox, L. P., Jung, J., McDaniel, P., & Sheth, A. N. (2014). TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. ACM Transactions on Computer Systems (TOCS), 32(2), 1–29.
2. Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). Android Permissions Demystified. Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS ’11), 627–638.
3. Au, K. W. Y., Zhou, Y., Huang, Z., & Lie, D. (2012). PScout: Analyzing the Android Permission Specification. ACM Conference on Computer and Communications Security, 217–228.
4. Zhou, Y., & Jiang, X. (2012). Dissecting Android Malware: Characterization and Evolution. IEEE Symposium on Security and Privacy, 95–109.
5. Iker Burguera, Urko Zurutuza, and Simin N. Tehrani. “Crowdroid: behavior-based malware detection system for Android”, In Proceedings of the 1st ACM workshop on Security and privacy in smartphones mobile devices, SPSM ’11, pages 15–26, New York, NY, USA, October 2011. ACM.
6. Shabtai A., Kanonov U., Elovici Y., Glezer C. and Y. Weiss, Andromaly: a behavioural malware detection framework for android devices, Journal of Intelligent Information Systems, 38(1), 161-190 (2012)
7. M. Zhao, F. Ge, T. Zhang, and Z. Yuan, “AntiMalDroid: An efficient SVM-based malware detection framework for android,” Commun. Comput. Inf. Sci., vol. 243 CCIS, pp. 158–166, 2011.

Received: August 09 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

8. W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," Osdi "10, vol. 49, pp. 1–6, 2010.
9. Zimperium. 2025 Global Mobile Threat Report. Apr 25, 2025.
10. Lu, H., et al. "Uncovering the App Cloud Access Risks under …" PoPETs / Cloudet (2025).
11. Milanese, R., et al. "Assessing the Effectiveness of an LLM-Based Permission Model for Android" (ScitePress, 2025).
12. Alkinoon, A. "A Comprehensive Analysis of Evolving Permission Usage" (MDPI Cyber, 2025).
13. Nguyen, T. T., et al. "Open Access Alert: Studying the Privacy Risks in Android …" (CISPA/2025).
14. Sabbah, A., Jarrar, R., Zein, S., Mohaisen, D. "Understanding Concept Drift with Deprecated Permissions in Android Malware Detection" (arXiv, Jul 29, 2025 — preprint dated 2025).
15. Zhang, X. "Understanding the Bad Development Practices of Android …" (IEEE/Computer Org., 2025).
16. Long, Y. "ARMOUR US: Android Runtime Zero-permission Sensor …" (University technical report, 2025).
17. Wu, X., et al. "Transparency or Information Overload? Evaluating Users' …" (NDSS/2025 technical report).
18. Wu, Y. "Modeling End-User Affective Discomfort With Mobile App …" (Georgia Tech tech report, 2025).
19. Paneva, V., et al. "User Understanding of Privacy Permissions in Mobile …" (arXiv/pdf, 2025).
20. Kouaho, Y. "A Study of User Awareness in Android App Permissions" (2025).
21. Gonde, D. "Mobile App Permissions and Data Privacy: Risks and …" (SSRN, 2025).
22. Lu, H., et al. (Cloudet paper — detailed cloud backend risks in mobile apps) (POPEtS/2025).
23. "Careless Errors in Hundreds of Apps Could Expose Troves of Data" — Symantec research coverage (2025 retrospective reporting).
24. Prasad, A. "AndroMD: An Android malware detection framework based on source code analysis and permission scanning" (SciDirect, 2025).
25. Haggag, O. "An analysis of privacy regulations and user concerns …" (ScienceDirect, 2025).
26. "Apple / CIS Security advisories: Multiple vulnerabilities in Apple products (permission/privilege implications)" (CISecurity, Jan 28, 2025).
27. TechGDPR. "Data protection digest 1–15 Jan 2025: mobile app permissions should work … CNIL" (Jan 17, 2025).
28. Bauer / CISPA research: "How wording influences consent behavior for app rationales" (CISPA, Apr 14, 2025).

Received: August 09 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

29. NowSecure (2025 research summary) — mobile app privacy risks and cloud AI endpoints (2025 summaries).
30. Bitdefender. “10 Cyberthreats iPhone Users Can't Afford to Ignore in 2025” (Feb 4, 2025) — includes permission/zero-click/privilege themes.
31. Tech news: “This new Android attack could trick you into compromising your own phone — TapTrap” (Tom’s Guide, 2025) — permission bypass via tapjacking research.
32. Wired / historical & 2025 coverage of app→cloud credential issues (reporting on cloud-backed mobile app risks) (2025).
33. Symmetrium. “Best Practices for Mobile Data Protection in 2025” (white paper, 2025) — permission/containment recommendations.
34. ICT/industry “2025 Security & Access Control Trends” (Feb 12, 2025) — access/permission trends for mobile & cloud. ICT - Integrated Control Technology
35. Communications-Co. “2025 Access Control Predictions: AI, the Cloud and Mobile …” Jan 2, 2025) — permission/credential observations.
36. “How Mobile Apps Are at Risk From Cloud Misconfigurations” — industry analysis (LinkedIn/Approov summary, 2025).
37. SentinelOne / cloud security statistics (2025) — cloud misconfiguration and permission missteps affecting mobile apps.
38. Wiz.io (2025 guidance) “Top 11 Cloud Security Vulnerabilities and How to Fix Them” — relevance to mobile app backend permissions.
39. ResearchGate: “Current Security Issues and Vulnerabilities Associated With Mobile Application” (2025 survey).
40. ResearchGate: “Abusing iOS Permissions: A Security Perspective” (2025).
41.. NowSecure / industry blog pieces (2025) — mobile apps leaking device/cloud data (summary).
42. “Mastering Android Permissions in 2025” — technical guide/blog (Jun 12, 2025) — practical permission model evolution.

Received: August 09 2025