iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.83288

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Early Stage Voice Phishing Detection with Runtime Permission Request: A Review

Ms. Aayushi Pardeshi[1], Ms. Rucha. R. Galgali[2]
[1, 2]*First-Third Department of Computer Science and Engineering, Deogiri Institute of Engineering and Management Studies, Chhatrapati Sambhajinagar, Maharashtra, 431005*

Abstract: ***Voice phishing (vishing) has emerged as one of the most damaging cyber threats targeting Android users globally, combining social engineering with specialized malware that exploits the device telephony stack. This paper reviews existing detection strategies for voice phishing malware on Android platforms and presents the design rationale, architecture, and preliminary results of VishielDroid — a machine learning-based Android application that monitors both install-time and runtime permission request patterns to detect vishing malware before malicious operations are executed. Drawing on analysis of 613 vishing malware samples and 1,248 benign applications (2021–2023), point-biserial correlation identified 98 statistically significant permission features. A Support Vector Machine (SVM) classifier achieves 99.86% accuracy, 100% precision, and 99.78% F1-score on unseen 2023 test samples. The system operates without root access or OS modifications, making it suitable for wide-scale deployment on standard consumer Android devices.***

Keywords: Voice Phishing, Vishing Detection, Android Security, Runtime Permissions, Machine Learning, SVM, VishielDroid, Malware Detection

I. INTRODUCTION

Android dominates the global mobile OS market, with over 70% share, making it a prime target for voice phishing, or vishing. This type of cybercrime uses voice communication and specialized malware to trick victims into giving up sensitive financial information or approving fake transactions. Vishing attacks have gotten a lot more sophisticated, with attackers using malicious Android apps to intercept calls, redirect communications, fake caller IDs, and record conversations without the victim even knowing.

There are two main ways to defend against vishing malware: static analysis and dynamic analysis. Static analysis looks at an app's permissions and code when it's installed, while dynamic analysis monitors what the app does while it's running. The problem is, static analysis can be evaded by code obfuscation and other tricks. Dynamic analysis, like HearMeOut, can detect attacks, but only during active phone calls, which may be too late.

The gap between when an attack starts and when it's detected basically is a major vulnerability. This review looks at how phishing threats have evolved, the Android permission model, and current detection methods. It also introduces VishielDroid, an early-stage detection system that tries to close this timing gap by analyzing runtime permission requests before any malicious activity happens.

II. LITERATURE REVIEW

An Research on phishing and Android malware detection has progressed considerably over the past decade. Key contributions are reviewed below.

A. Evolution of Phishing and Vishing

Phishing began as mass email fraud in the 1990s and has expanded to spear-phishing, SMS-based smishing, and eventually voice-based vishing. Song et al. (2014) have previously shown caller ID spoofing to be successful in over 80% of cases under laboratory conditions, thus providing the root threat of telephony impersonation. [3]

Sahin et al. (2017) provided a comprehensive survey of telephony fraud networks, in which they analyzed and identified some of the technical infrastructure used in bulk vishing attacks, while also identifying the deficiency in present defenses. [4]

Nahapetyan et al. (2024) studied SMS phishing infrastructure and tactics and found multi-channel attacks using a combination of SMS to draw victim to a malicious APK download that bridges smishing with vishing. [11]

©IJRASET: All Rights are Reserved

B. Android Permission Model and Runtime Permissions

Android 6.0 (API level 23) introduced the runtime permission model, where the granting of high-risk functionalities (sensitive) occurs at the time of their use rather than at installation time. Though a user-friendly mechanism, this opening was exploited by vishing malware through which the requested high risk permission could be asked to be granted only at post installation time through UI overlay and social engineering (e.g. During the vishing call scenario in progress). Li et al. [8] showed that classification using permissions results in 85-90% accuracy for the traditional malware classes.

C. Static and Dynamic Malware Detection

Wu et al (2019) introduced MalScan that used the concept of social-network centrality analysis of API call graphs to enable rapid, market-wide scanning for malware, but faced challenges with its high dimensional feature space and computational cost that are impractical for an on-device scanner. [5]

Li et al (2021) explored adverserial robustness of android malware detection and developed defenses against attacks in feature space. These analyses showed the importance of an evasion-proof detection mechanism. [6]

Singh et al (2024) developed an interpretable dynamic analysis that inspected API call traces at runtime. Despite the improved interpretability this method requires a long execution time and is not feasible to be run on a device. [7]

Onwuzurike et al (2019) introduced MaMaDroid that modeled API call behavior using Markov chains and reported high detection rates, but requires a high computational cost and did not focus on vishing behavior. [9]

D. Vishing-Specific Detection

Kim et al. (2022) presented HearMeOut, which customizes Android OS, observing telephony API call during active call to determine call redirection, caller-id spoofs and pre-recorded sounds usage. Although it can identify attack pattern that are already known and detectable during the ongoing of the attack, OS level modification restricts its large scale application.[1]

Lee et al. (2025) performed the foundational empirical study closest to VishielDroid, calculating the point-biserial correlation between 246 permission-timing combinations (123 permission at installation time and 123 permission at runtime). They discover 98 significant features (60 at installation time, 38 at runtime) and they found WRITECALLLOG (r=0.975), PROCESSOUTGOINGCALLS (r=0.971), READSMS (r=0.969), READCALL_LOG (r=0.965) have strong positive correlations with vishing malware.[2]

E. Feature Selection and Machine Learning

Kouliaridis et al (2020) conducted a study on the feature importance on Android Malware classification and observed that the permission based features have high discrimination power as well as efficient for on device classification as they are less resource consuming. [10]

It has been shown in multiple comparisons between classifiers (SVM, Random Forest, KNN, Logistic Regression and neural networks) that SVM with a linear kernel significantly outperforms other classifiers for high dimensional binary feature spaces like permission vectors due to high margin maximization and over fitting avoidance on sparse feature space.it.

III. REASEARCH GAP

From the literature review, we identified the following five open research questions that motivate the proposed VishielDroid:

1) Early-Stage Detection Gap: None of the previous studies integrate the prediction of attack execution before it happens and the runtime behavior-based permission pattern analysis.

2) Runtime Permission Analysis Gap: Although many existing works studied permissions, the temporal information carried by permission requests at runtime hasn't been sufficiently investigated as a primary classifier of malwares by itself.

3) Vishing-Specific Gap: Many malware detection models were designed for more general malware families, not specifically for vishing attacks. Often, the evaluation was done with generalized malware datasets instead of ones focusing on telephony-based attacks.

4) Practical Deployment Gap: We identified that approaches which modify OS (e.g., HearMeOut) or require high-dimensional models (e.g., MalScan) limit deployment capabilities without device rooted access or system changes.

5) Hybrid Analysis Gap: To the best of our knowledge, no previously deployed solutions combined both install-time and runtime permission features into a single, low-footprint on-device classifier for vishing detection.

IV. PROPOSED SYSTEM: VISHIELDROID

A. *System architecture*

VishielDroid is a conventional Android application (min SDK: API 23) which aims to detect vishing malware at the privilege escalation phase (post-installation and pre-malicious telephony operations). It does not need root privilege, nor OS/firmware customization. This permits deployment on a wide spectrum of consumer Android devices (Android 6.0 to 12).

B. *Three-Phase Attack Model and Detection Strategy*

Vishing malware involves a three stage attack process: (1) Installation of the software using non-official channel likeSMSlink or third party stores. (2) privilege escalation using runtime permission requests which would grant the malware access to telephony services. (3) deceptive actions like intercepting and rerouting calls while there's an ongoing call. VishielDroid mainly focus on stage 2 as the target of attack and try to detect the malware before its operation of phase 3 will cause damage to the victim.

C. *Architecture Components*

The system includes four modules: (1)Permission Monitoring Module (PMM)-monitor installations and runtime permission requests by broadcast receiver; (2) Feature Extraction Module (FEM)-generate binary feature vectors from install-time and run-time permissions; (3) Classification Module (CM)-use a pre-trained SVM classifier with linear kernel to predict threat scores; and (4) User Interface and Alert Module (UIAM)-give a context alert and removal malware option before malware execution.

D. *Feature Engineering*

From the analysis, the point-biserial correlation over 613 vishing samples and 1248 benign apps yielded 98 permissions to have statistically significant discriminative powers ($p<0.1$). Features are encoded as binary indicators and contain 60 install-time and 38 runtime permission indicators. This binary encoding neither requires any feature scaling and is sparse so that can be efficiently computed. Top critical correlated permissions to vishing applications: WRITECALLLOG (r=0.975), PROCESSOUTGOINGCALLS (r=0.971), READSMS (r=0.969), READCALLLOG (r=0.965) and WRITECONTACTS (r=0.913).

E. *Machine Learning Model*

A linear SVM classifier is trained in Python (scikit-learn) offline, then exported to ONNX and used to run inference on device using ONNX Runtime for Android. The size of the model is 2.93 KB, classification latency < 1ms, the memory requirement is less than 40MB and daily battery drain is less than 2% (parameters checked from Android 8.1 up to 12).

V. RESULTS AND PERFORMANCE EVALUATION

VishielDroid was evaluated against unseen 2023 test samples not included in the training set, providing an honest assessment of generalization. Table I summarizes key performance metrics.

TABLE I
VISHIELDROID PERFORMANCE METRICS ON UNSEEN 2023 TEST SAMPLES

Metric	VishielDroid (SVM)	Best Comparable System
Accuracy	99.86%	~80.33%
Precision	100.00%	~91.20%
Recall	99.57%	~88.70%
F1-Score	99.78%	~80.25%
F1-Score (Class Imbalance)	97.78%	N/A
Model Size	2.93 KB	—
Classification Latency	< 1 ms	—
Memory Footprint	< 40 MB	—
Daily Battery Consumption	< 2%	—

The F1-score performance of VishielDroid is 19.53 points higher than that of the best comparable system. Even with high class imbalance (1:10 vishing vs benign ratio), VishielDroid can perform well with F1-score 97.78%. This suggests that it can adapt to real world environments where benign apps significantly outnumber malicious ones.

©IJRASET: All Rights are Reserved

VI. CONCLUSION

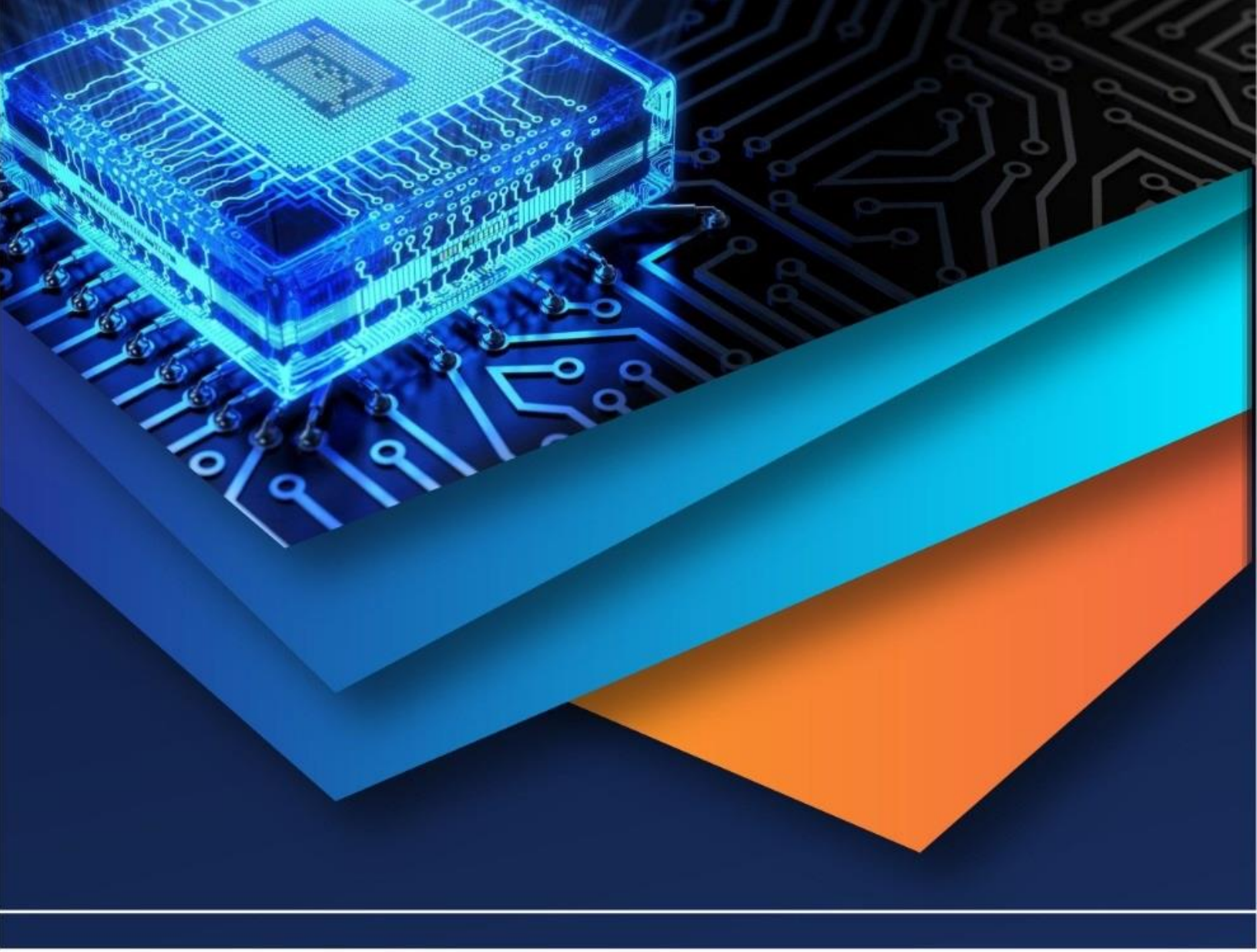
This paper traced the evolution of voice phishing threats and discussed current approaches for detecting Android malware, exposing a need for detection before harmful actions are carried out, runtime behavioral analysis, and easy-to-deploy solutions. VishielDroid meets this need by utilizing a hybrid permission analysis-a combination of install time declarations and runtime request patterns-with a light-weight, offline, rooted or unrooted SVM toclassify requests.

VishielDroid's second phase strategy generates proactive alerts to phone call-related activity before any malicious calls can occur, which is a qualitative improvement over previously existing reactivesolutions such as HearMeOut. With its extremely light weight size(2.93KB model, <1ms delay, <2% battery usage per day), it is capable of being deployed to mass consumer devices running Android versions 8.1 to 12. Future work is expected to look into using LSTM's for temporal sequences, fusing additional multi-modal features such as API call patterns and network activity, federated learning to provide model updates in a private way, and adding detections for related types of malware, such as banking Trojans and spy ware.

REFERENCES

[1] Kim, J., Kim, J., Wi, S., Kim, Y., & Son, S. (2022). HearMeOut: Detecting voice phishing activities in Android. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (pp. 289–301).
[2] Lee, C., Kim, B., & Kim, H. (2025). The silence of the phishers: Early-stage voice phishing detection with runtime permission requests. Computers & Security, 152, 104364.
[3] Song, J., Kim, H., & Gkelias, A. (2014). iVisher: Real-time detection of caller ID spoofing. ETRI Journal, 36(5), 865–875.
[4] Sahin, M., Francillon, A., Gupta, P., & Ahamad, M. (2017). SoK: Fraud in telephony networks. In Proceedings of the 2nd European Symposium on Security and Privacy (pp. 235–250).
[5] Wu, Y., Li, X., Zou, D., Yang, W., Zhang, X., & Jin, H. (2019). MalScan: Fast market-wide mobile malware scanning by social-network centrality analysis. In Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (pp. 139–150).
[6] Li, H., Zhou, S., Yuan, W., Luo, X., Gao, C., & Chen, S. (2021). Robust Android malware detection against adversarial example attacks. In Proceedings of the 30th International Conference on World Wide Web (pp. 3603–3612).
[7] Singh, A., Tanha, M., Girdhar, Y., & Hunter, A. (2024). Interpretable Android malware detection based on dynamic analysis. In Proceedings of the 10th International Conference on Information Systems Security and Privacy.
[8] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., & Ye, H. (2018). Significant permission identification for machine-learning-based Android malware detection. IEEE Transactions on Industrial Informatics, 14(7), 3216–3225.
[9] Onwuzurike, L., Mariconti, E., Andriotis, P., Cristofaro, E. D., Ross, G., & Stringhini, G. (2019). MaMaDroid: Detecting Android malware by building Markov chains of behavioral models (Extended version). ACM Transactions on Privacy and Security, 22(2), 1–34.
[10] Kouliaridis, V., Kambourakis, G., & Peng, T. (2020). Feature importance in Android malware detection. In Proceedings of the 19th International Conference on Trust, Security and Privacy in Computing and Communications.
[11] Nahapetyan, A., Prasad, S., Childs, K., Oest, A., Ladwig, Y., Kapravelos, A., & Reaves, B. (2024). On SMS phishing tactics and infrastructure. In Proceedings of the 45th IEEE Symposium on Security and Privacy.
[12] Android Developers. (2024). Android 6.0 changes. Retrieved from https://developer.android.com/about/versions/marshmallow/android-6.0-changes

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)