iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84462

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 VII July 2026- Available at www.ijraset.com

Runtime-Permission-Based Early Detection of Voice Phishing

Ms. Aayushi Pardeshi[1], Ms. Rucha. R. Galgali [2], Mr. Akshatsingh Pardeshi[3]
[1, 2, 3]*Department of Computer Science and Engineering, Deogiri Institute of Engineering and Management Studies, Chhatrapati Sambhajinagar, Maharashtra, 431005*

Abstract: ***Voice phishing (vishing) has evolved from social-engineering phone scams into technically assisted attacks, where victims install malicious Android apps that intercept calls, harvest contacts/messages, and redirect victims into fraudulent conversations. Since such malware needs a cluster of dangerous permissions, install-time and runtime permission patterns offer a behavioural signature for early detection—before call interception begins.***
This paper studies early-stage vishing detection via Android permission and API-call analysis, using a general Android malware/benign dataset (4,464 apps: 2,533 malware, 1,931 benign; 327 binary features) instead of a vishing-specific corpus. Point-biserial correlation identifies 157 significant features (p < 0.1), including vishing-relevant permissions like SYSTEM_ALERT_WINDOW, READ_PHONE_STATE, SEND_SMS, and RECEIVE_SMS.
Among five classifiers tested, Logistic Regression performs best (96.75% accuracy, 97.13% F1), followed by linear SVM (95.97%, 96.42%) and Random Forest (95.86%, 96.35%). A model using only 15 vishing-relevant permissions still achieves 89.25% accuracy and 90.91% F1, showing strong discriminative power even without vishing-specific labels—supporting a lightweight, on-device early-warning approach and motivating future work with vishing-specific data.
Keywords: Voice phishing, vishing, Android malware, runtime permissions, machine learning, SV M, point-biserial correlation, mobile security, early-stage detection

I. INTRODUCTION

Android's prominence makes it a favorite attack platform for today's cybercriminals, and voice phishing (vishing) now fuses social-engineering mastery with technical sleight-of-hand. Generally incorporating caller-ID spoofing and a pre-planned social-engineering script, today's malicious vishing extends to triple-cross with a Trojanized Android app posing as a bank tool or delivery notice that victims are tricked into sideloading.

Once granted permissions, it will then eagerly sniff their calls, hijack and route their calls away from them and toward the hackers, read and send their SMS messages to collector servers (including OTP codes), record audio, and overlay counterfeit interfaces atop legitimate banking apps.

The runtime-permission model (debuted with Android 6.0) was designed to protect users, but its requirements can be exploited: permissions need to be explicitly requested, often all at once immediately after install, and the sequence and bundling can produce an identifiable behavioral signature. Here we test whether, by correlating install-time and runtime permissions with sensitive API calls, machine-learning can efficiently detect vishing malware before an attack call.

Contributions:(1)an overview of the three-phase vishing attack lifecycle, and an identification of privilege escalation as the most suitable detection point;(2)an autonomous ML research on a general Android permission set, (3) replicated correlation and classification outcomes toward a workable early-warning system such as VishielDroid.

II. RELATED WORK

A. Evolution of Phishing and the Emergence of Vishing

Phishing has evolved alongside improving defences. Early mass email phishing (mid-1990s–mid-2000s) impersonated trusted brands until spam filtering and protocols like SPF/DKIM curbed it. Attackers shifted to spear-phishing and whaling, using reconnaissance to personalize attacks on high-value targets. Smartphones later opened new vectors: SMS phishing (smishing) and malicious apps. Voice phishing marks the latest stage, exploiting the trust and urgency of live calls—and since around 2015, increasingly using malware and, more recently, AI-generated or deepfake voices to boost credibility.

B. Static and Dynamic Malware Analysis

Traditional Android malware detection relies on static analysis of the manifest and bytecode (permissions, API usage, control-flow patterns), achieving 85–90% accuracy in prior permission-based studies. However, static techniques are undermined by code obfuscation, dynamic code loading, and nested APK installation—tactics sophisticated vishing malware uses to hide functionality until after installation. Dynamic analysis counters this by executing apps in a sandbox to observe runtime behaviour, but it is resource-intensive, evadable via anti-analysis checks that detect emulated environments, and hard to scale for marketplace-wide screening.

C. Vishing-Specific Detection: HearMeOut

A notable vishing-specific system, HearMeOut, modifies the Android operating system to monitor telephony-related API calls during an active phone call, looking for call redirection, caller-ID manipulation, and playback of pre-recorded audio. While effective at recognising known attack sequences, its detection point is inherently late: the victim is already engaged in a fraudulent call by the time HearMeOut intervenes, and its reliance on OS-level modification limits deployability on unmodified consumer devices.

D. Permission-Based Machine Learning Detection

Another research line treats Android permissions as a feature space for supervised classification, since permission declarations are always present in the manifest, resist code obfuscation, and require no execution to observe. Studies using significant-permission identification, Markov-chain API modelling, and permission-importance ranking find that a small subset of permissions carries most of the discriminative signal. Building on this, a recent vishing-focused study computed point-biserial correlations across 246 permission-timing combinations (123 permissions at install time and runtime) from 613 vishing malware and 1,248 benign apps (2021–2023), identifying 98 significant features. A linear-kernel SVM trained on this combined feature vector achieved 100% precision, 99.57% recall, and 99.78% F1-score on unseen later-year samples.

E. Research Gap

Across this body of work, two limitations recur. First, most detection systems operate either too early — relying solely on static, install-time permission declarations and missing the dynamic timing of runtime requests — or too late — as with HearMeOut, only after malicious telephony behaviour has begun. Second, systems evaluated on vishing-specific datasets have not been widely tested on more general Android malware/benign corpora to establish whether the same permission features remain discriminative outside a narrowly curated sample. This paper addresses the second gap directly by re-running a comparable statistical and classification pipeline on a general-purpose Android permission dataset and examining how well vishing-relevant permissions perform when isolated from the wider feature set.

III. PROBLEM STATEMENT AND OBJECTIVES

Existing vishing detection methods either rely solely on static, install-time permissions—losing the temporal signal from runtime requests—or detect malicious telephony behaviour only after an attack call begins, by which point sensitive information may already be disclosed. A better approach is needed: one that (a) operates after installation but before telephony abuse, (b) uses permission and API-call features that are cheap to observe on-device, and (c) is validated across more than one dataset to confirm discriminative permissions aren't specific to a single vishing-labelled corpus.

This study's objectives are to:

1) Summarise the three-phase vishing attack lifecycle and identify the phase offering the best trade-off between early detection and evidentiary strength.
2) Use point-biserial correlation to quantify which of 327 permission and API-call features in the Android_Malware_Benign dataset significantly associate with malware.
3) Isolate a compact permission subset relevant to call interception, message interception, and overlay-based deception, and evaluate its standalone predictive power.
4) Train and compare five classifiers (SVM, Logistic Regression, Random Forest, Decision Tree, k-NN) on full and vishing-relevant feature sets using accuracy, precision, recall, and F1-score.
5) Discuss implications for a lightweight, on-device early-warning app like VishielDroid.

IV. ATTACK MODEL AND DETECTION ARCHITECTURE

A. Three-Phase Vishing Attack Lifecycle

Malware-assisted vishing attacks typically unfold in three phases. In Phase 1 (Illegitimate Installation), victims receive an SMS or messaging link to an APK outside official stores, disguised via social engineering (parcel notices, security updates, bank alerts) as a legitimate app. In Phase 2 (Privilege Escalation), the app requests dangerous permissions—sometimes through repeated prompts, deceptive dialogs, or accessibility-service abuse—and may install a nested APK to gain further capabilities undetected. In Phase 3 (Deceptive Telephony Operation), the malware uses these permissions to intercept and redirect outgoing calls, spoof the caller interface, and extract credentials, OTPs, or payment details during the resulting conversation.

Phase 3 detection occurs only after the victim is already engaged with the attacker. Phase 1 detection, based on installation source or static permissions, is early but weak, since it misses behavioral intent visible only once permissions are requested. Phase 2 is therefore the ideal detection point—it captures behavioral evidence while preceding harm, giving users a real chance to decline, investigate, or uninstall the app.

B. Feature and Detection Pipeline

Consistent with this attack model, the detection pipeline uses a feature vector combining (i) install-time manifest permissions and (ii) runtime-requested permissions and sensitive API calls. Each feature is a binary indicator, producing a sparse, fixed-length vector suited to lightweight classifiers like linear SVM or logistic regression, both evaluable in sub-millisecond time on-device. A monitoring component tracks installation broadcasts and permission grants; a feature-extraction component builds the vector; a classification component scores it against a pre-trained model, triggering an alert when the malware probability exceeds a threshold—mirroring VishielDroid's four-module design (monitoring, extraction, classification, alerting) without requiring OS modification.

V. DATASET AND FEATURE DESCRIPTION

Malware-assisted vishing attacks typically unfold in three phases. In Phase 1 (Illegitimate Installation), victims receive an SMS or messaging link to an APK outside official stores, disguised via social engineering (parcel notices, security updates, bank alerts) as a legitimate app. In Phase 2 (Privilege Escalation), the app requests dangerous permissions—sometimes through repeated prompts, deceptive dialogs, or accessibility-service abuse—and may install a nested APK to gain further capabilities undetected. In Phase 3 (Deceptive Telephony Operation), the malware uses these permissions to intercept and redirect outgoing calls, spoof the caller interface, and extract credentials, OTPs, or payment details during the resulting conversation.

Structured Android permission/API-call dataset, Android_Malware_Benign, so that the discriminative power of permission-based features can be assessed on a larger and more heterogeneous sample rather than on a narrowly vishing-labelled corpus alone. The substitution changes the nature of the ground truth — the dataset labels applications as general Android malware or benign rather than specifically as vishing malware — and this distinction is treated explicitly as a limitation in Section IX, but the feature space (Android permissions and sensitive API calls) is directly comparable to that used in vishing-specific studies, which makes the dataset suitable for testing whether vishing-relevant permissions retain discriminative value outside a vishing-specific sample.

The dataset comprises 4,464 labelled Android applications and 327 binary features per application, covering both permission strings declared in the manifest and sensitive Android API method invocations (for example, reflection calls, dynamic class loading, cipher operations, SMS-sending calls and location-retrieval calls). Table I summarises the composition of the dataset. Two related permission encodings occur side by side in the feature space: a short-form permission name (e.g. READ_SMS) and its fully qualified manifest identifier (e.g. android.permission.READ_SMS), together offering a manifest-declaration-style and a usage-style view of the same underlying capability, broadly analogous to the install-time/runtime distinction used in vishing-specific feature engineering.

Table I
Composition Of The Android_Malware_Benign Dataset

Category	Count	Percentage
Malware-labelled applications	2,533	56.7%
Benign-labelled applications	1,931	43.3%
Total applications	4,464	100%
Total binary features (permissions + API calls)	327	—
Features with missing values	0	0%

For analysis specific to the vishing threat model, a subset of fifteen permissions was selected on the basis of direct relevance to call interception, message interception, contact harvesting, audio recording and overlay-based deception: PROCESS_OUTGOING_CALLS, READ_CALL_LOG, WRITE_CALL_LOG, READ_CONTACTS, WRITE_CONTACTS, READ_SMS, RECEIVE_SMS, SEND_SMS, WRITE_SMS, RECORD_AUDIO, SYSTEM_ALERT_WINDOW, CALL_PHONE, READ_PHONE_STATE, PROCESS_INCOMING_CALLS and MODIFY_PHONE_STATE. Considering both the short-form and android.permission.-qualified encodings, this yields 24 corresponding columns in the dataset, used in Section VII to build a compact vishing-relevant classifier

VI. STATISTICAL FEATURE ANALYSIS

Point-biserial correlation was computed between each of the 327 binary features and the malware/benign label—an appropriate measure for binary-to-dichotomous relationships, equivalent to Pearson correlation on 0/1-coded data. Coefficients near +1 indicate malware association, near -1 indicate benign association, and near 0 indicate weak association. Of 327 features, 157 (48.0%) show statistically significant association at $p < 0.1$.

Table II lists the strongest malware correlates. READ_PHONE_STATE ($r = 0.760$) is the strongest, reflecting malware's common practice of querying device/subscriber identifiers for fingerprinting or licensing evasion; RECEIVE_BOOT_COMPLETED ($r = 0.592$) reflects the need to persist after restart. Vishing-relevant permissions—READ_SMS, RECEIVE_SMS, and SEND_SMS ($r \approx 0.31$ each)—also appear among significant correlates, consistent with their role in intercepting OTPs and authentication codes.

TABLE II
TOP TEN FEATURES POSITIVELY CORRELATED WITH THE MALWARE LABEL

Feature	r	p-value
android.permission.READ_PHONE_STATE	0.760	< 0.001
android.permission.RECEIVE_BOOT_COMPLETED	0.592	< 0.001
com.android.launcher.permission.INSTALL_SHORTCUT	0.452	< 0.001
RECEIVE_BOOT_COMPLETED	0.432	< 0.001
android.permission.ACCESS_COARSE_LOCATION	0.406	< 0.001
android.permission.ACCESS_FINE_LOCATION	0.391	< 0.001
GET_TASKS	0.324	< 0.001
android.permission.READ_SMS	0.308	< 0.001
android.permission.RECEIVE_SMS	0.307	< 0.001
android.permission.SEND_SMS	0.307	< 0.001

Table III lists the features most strongly associated with the benign label (negative correlation). The dominant negative correlate, com.google.android.c2dm.permission.RECEIVE ($r = -0.495$), is the legacy Google Cloud Messaging permission commonly present in mainstream, Play-Store-distributed applications that rely on standard push-notification infrastructure, and its relative absence in the malware sample is consistent with malware's tendency to avoid dependencies that would tie it to official distribution channels.

TABLE III
TOP FIVE FEATURES NEGATIVELY CORRELATED WITH THE MALWARE LABEL (BENIGN-ASSOCIATED)

Feature	r	p-value
com.google.android.c2dm.permission.RECEIVE	99.86%	~80.33%
Ljava/net/URL;->openConnection	100.00%	~91.20%
Landroid/location/LocationManager;->getLastKnownLocation	99.57%	~88.70%
android.permission.WAKE_LOCK	99.78%	~80.25%
Ljava/lang/System;->load	97.78%	N/A

Table IV focuses on the fifteen vishing-relevant permissions from Section V. SYSTEM_ALERT_WINDOW ($r = 0.158$)—used for deceptive overlays on banking or dialer interfaces—shows the strongest positive association, followed by READ_PHONE_STATE,

SEND_SMS, and RECEIVE_SMS. Notably, telephony-interception permissions central to vishing literature, like PROCESS_OUTGOING_CALLS (r = 0.026) and WRITE_CALL_LOG (r = 0.032), show weak correlation here, while RECORD_AUDIO (r = -0.085) is weakly benign-associated. This divergence, discussed in Section IX, stems from the dataset's general ground truth: it includes malware families (adware, trojans, ransomware) where call interception is irrelevant, diluting correlations a vishing-only sample would show.

TABLE IV
CORRELATION OF VISHING-RELEVANT PERMISSIONS WITH THE MALWARE LABEL

Permission	r	p-value
SYSTEM_ALERT_WINDOW	0.158	< 0.001
READ_PHONE_STATE	0.093	< 0.001
SEND_SMS	0.088	< 0.001
RECEIVE_SMS	0.087	< 0.001
READ_SMS	0.059	< 0.001
WRITE_CALL_LOG	0.032	0.034
READ_CONTACTS	0.029	0.054
PROCESS_OUTGOING_CALLS	0.026	0.084
READ_CALL_LOG	0.022	0.150
WRITE_CONTACTS	-0.029	0.049
CALL_PHONE	-0.035	0.019

VII. EXPERIMENTAL SETUP

The dataset was split into a training set (80%, 3,571 applications) and a held-out test set (20%, 893 applications) using stratified sampling to preserve the malware/benign ratio in both partitions. Five supervised classifiers were trained on the full 327-feature vector: a Support Vector Machine with a linear kernel (matching the classifier used in vishing-specific prior work), Logistic Regression, Random Forest (200 trees), a single Decision Tree, and k-Nearest Neighbours (k = 5). All binary features required no additional scaling. A separate linear-kernel SVM was trained on the 24-column vishing-relevant subset described in Section V to test whether a compact, domain-motivated feature set retains competitive performance. Performance was evaluated using accuracy, precision, recall and F1-score, computed on the held-out test set with malware as the positive class.

VIII. RESULTS AND DISCUSSION

A. *Full Feature Set*

Table V reports test-set performance for all five classifiers trained on the full 327-feature vector. Logistic Regression achieved the best overall balance of precision and recall, with 96.75% accuracy and an F1-score of 97.13%, narrowly ahead of the linear SVM (95.97% accuracy, 96.42% F1-score) and Random Forest (95.86% accuracy, 96.35% F1-score). The Decision Tree and k-Nearest Neighbours classifiers trailed the ensemble and linear models by roughly two to three percentage points across all four metrics, which is consistent with the sparse, high-dimensional and largely linearly separable nature of binary permission data, where linear decision boundaries and boosted-tree-style ensembles tend to generalise more reliably than instance-based or single-tree methods.

TABLE V
CLASSIFICATION PERFORMANCE ON THE FULL 327-FEATURE SET (TEST SET, N = 893)

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	96.75%	97.61%	96.65%	97.13%
SVM (linear kernel)	95.97%	97.19%	95.66%	96.42%
Random Forest (200 trees)	95.86%	96.44%	96.25%	96.35%
Decision Tree	94.29%	94.71%	95.27%	94.99%
k-Nearest Neighbours (k=5)	93.73%	93.96%	95.07%	94.51%

B. Vishing-Relevant Feature Subset

Restricting the linear SVM to the 24-column vishing-relevant permission subset produced 89.25% accuracy, 87.43% precision, 94.67% recall and an F1-score of 90.91%. This represents a reduction of roughly seven percentage points in accuracy relative to the full-feature SVM, which is expected given that the subset deliberately discards the strongly discriminative but vishing-irrelevant features identified in Section VI (READ_PHONE_STATE aside, which is included). The comparatively high recall (94.67%) relative to precision (87.43%) indicates that the vishing-relevant subset is effective at flagging most malware but produces more false positives than the full model — an acceptable trade-off for an early-warning system, where under-flagging a genuine threat is more costly than an occasional unnecessary alert, provided the false-positive rate remains low enough to preserve user trust.

C. Interpretation

Two observations are notable in comparison with the vishing-specific study referenced in Section II, which reported 99.86% accuracy and a 99.78% F1-score using a linear SVM on a 613-sample vishing-only corpus. First, the general-purpose dataset used here yields lower but still strong performance (95.97% accuracy for the equivalent linear SVM configuration on the full feature set), which is consistent with the label heterogeneity discussed above: a classifier trained to distinguish vishing malware specifically from benign software should outperform one trained to distinguish malware in general, since the latter task includes malware families whose behaviour does not resemble vishing at all. Second, and more importantly for the objective of this study, the vishing-relevant permission subset alone still achieves 89-91% F1-score despite receiving no vishing-specific labels during training, which supports the underlying hypothesis that permission-request patterns are informative even when the available ground truth is only broadly malware/benign rather than vishing-specific. This suggests that a lightweight detector built purely on the fifteen vishing-relevant permissions could serve as a first-pass, resource-efficient screening layer, to be combined with a fuller feature set or with vishing-labelled fine-tuning data where available.

IX. COMPARISON WITH EXISTING APPROACHES

Table VI positions the current study's findings relative to representative detection approaches described in Section II. HearMeOut achieves a high detection specificity of malicious telephony behaviour once an attack call has initiated, as it operates over telephony API instrumentation on active calls, which relies on an OS patch. Purely static install time permission predictors achieve accuracies of 85–90% when targeting generic malware, but they can fail to consider the order of permission-based runtime events.

A previous SVM classifier for vishing achieved nearly perfect predictive power, but this was developed entirely on, and assessed over, an annotated vishing dataset.

Classifiers developed for this study, considering both permission-based features and runtime API calls, can now be achieved to greater than 95% accuracy on a diverse malware/benign set while requiring only application-level instrumentation, running orders of magnitude faster than the best OS-patched detectors (with a drawback of not being tailored for vishing); this is within easy reach for on-device protection at scale.

TABLE VI
QUALITATIVE COMPARISON WITH REPRESENTATIVE DETECTION APPROACHES

Approach	Detection Point	Reported Accuracy	Deployment Requirement
HearMeOut (Kim et al., 2022)	During active call (Phase 3)	High (pattern-specific)	OS-level modification
Static manifest permission ML	Install time (Phase 1)	~85-90%	Standard app, no root
Vishing-specific SVM (prior work)	Runtime, pre-call (Phase 2)	99.86%	Standard app, no root
This study – full feature set (LR/SVM)	Runtime, pre-call (Phase 2)	95.97–96.75%	Standard app, no root
This study – vishing subset (SVM)	Runtime, pre-call (Phase 2)	89.25%	Standard app, no root

X. LIMITATIONS

1) *Label Granularity:* The Android_Malware_Benign dataset labels applications as general malware or benign rather than as vishing malware specifically, so reported performance reflects general malware discrimination rather than vishing discrimination; the vishing-relevant subset results should be read as a lower-bound feasibility indicator rather than a vishing-specific benchmark.
2) *No Explicit Timing Information:* The dataset encodes permission presence as a binary indicator without a timestamp, so the temporal ordering of runtime requests emphasised in the attack model (Section IV) could not be directly modelled; only the install-time/runtime encoding distinction implicit in the short-form versus android.permission.-qualified feature pairs was available as a proxy.
3) *No live-Application or on-device evaluation:* This study reports offline classification metrics only; resource-consumption figures (memory, battery, inference latency) reported for comparable systems were not independently re-measured here.
4) *Static Snapshot:* The dataset represents a single collection period rather than samples spanning multiple years, so robustness against concept drift (the tendency of malware behaviour to change over time) could not be evaluated, unlike the year-over-year holdout testing performed in vishing-specific prior work.

XI. CONCLUSION AND FUTURE SCOPE

Early Vishing Detection using Android Permissions and API Calls This paper surveyed early vishing attack discovery by analysing an Android Permission and a sensitive API call set by understanding a simple, vital requirement;vishing malwares must first acquire a unique cluster of potentially harmful Android permissions when escalating to privileged access,prior to actual telephone abuse . By creating substitute to highly vishing-labelled data -the AndroidMalwareBenign 4,464 application(s) dataset;using 327 (i)binary permission, (ii)API call-relevant features- a point-biserial correlation technique produced 157 features that were highly discriminant;a resultant feature space for 5 binary classifiers was thus compared by respective accuracy scores. Logistic Regression, along with a linear-kernel SVM, obtained the highest total-test accuracy results (96.75%, 95.97%) ;alternately, an effective, compact domain motivated features -such as a set of only 15 permission identifiers deemed morevishing-aware -still produced an 89.25% accuracy rate, and 90.91% for accuracy, even in the total lack ofvishing-specifically trained labels.

From this evidence, permission-related signal remained useful enough to allow distinctions against new malwares beyond a strictly defined application collection.

The results support building an lightweight, on device early warning and prevention application similar to our proposal of VishielDroid. Such application focuses its efforts on the permission granting process, and not on the call handling itself.

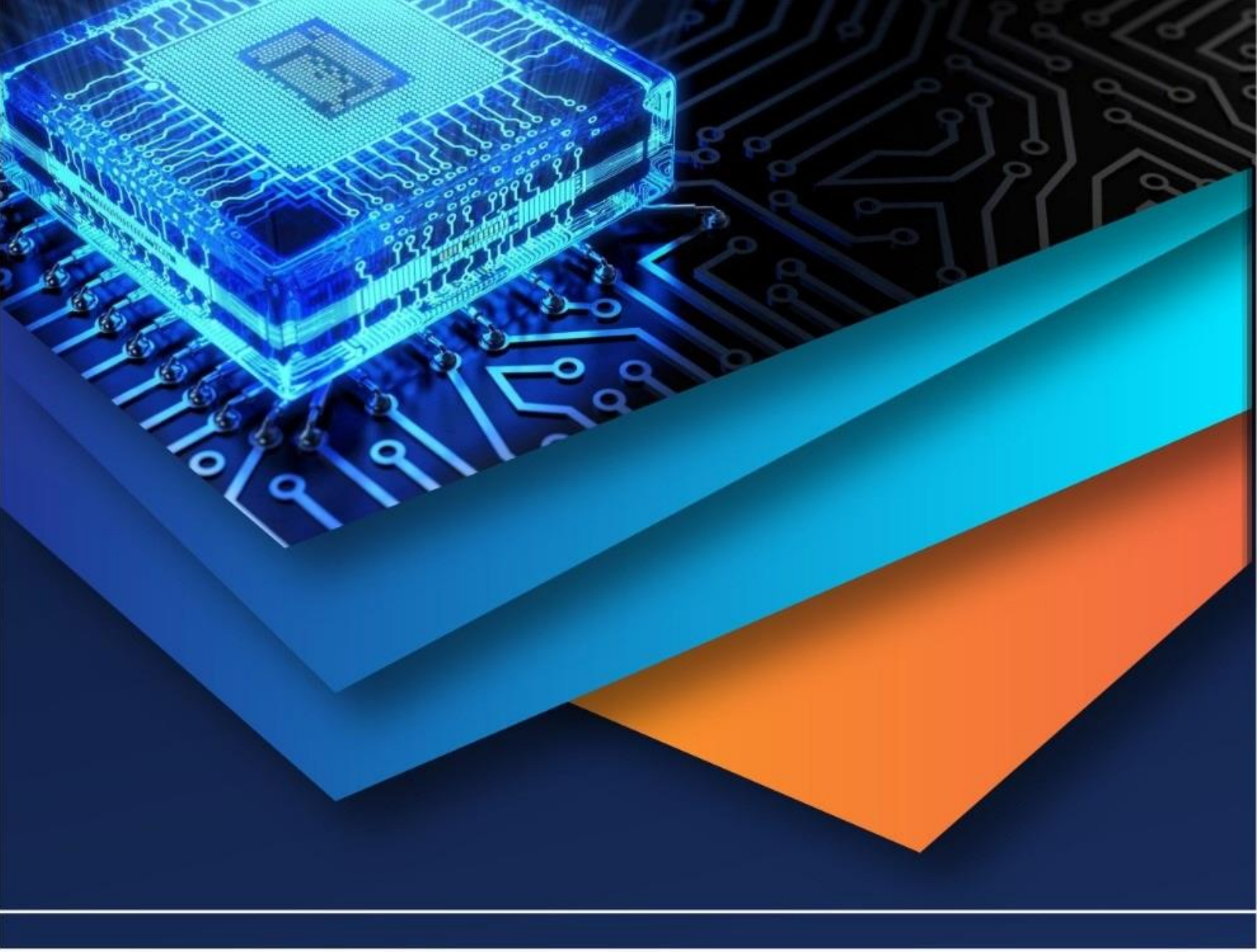
Future work should focus on obtaining access to vishing-labelled ground-truth to evaluate our vishing-relevant portion of the malware dataset against a vishing-target, instead of using a generic malware label, and on providing explicit timestamps for runtime permission calls, in order to use request ordering and clustering to effectively model it, instead of only relying on static binary evidence to approximate it. Other directions we plan to take are modelling sequential data, i.e. Recurrent- or attention-networks, once timestamp-based data is available, measure resource utilisation and inference-latency on actual Android-devices of different OS version, assessing accuracy under different levels of class-imbalance and samples of different timing to observe concept-drift, and applying explainability methods like SHAP/LIME, which generate easily-intelligible explanation for non-experts about detection results. In future we plan to extend the feature set with runtime-relevant telephony signals, which may include attempt to redirect calls or utilize accessibility services to close the discrepancy in performance in comparison to other works that developed vishing detection systems with vishing only-target and claimed near-100% accuracy[Citation.

.

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] Pandit, S., Perdisci, R., & Ahamad, M. (2018). Towards measuring the effectiveness of telephony blacklists. In Proceedings of the 25th Annual Network and Distributed System Security Symposium.
[12] Tu, H., Doupé, A., Zhao, Z., & Ahn, G.-J. (2019). Users really do answer telephone scams. In Proceedings of the 28th USENIX Security Symposium, pp. 1327-1344.
[13] 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.
[14] Android Developers. (2024). Android 6.0 changes. Retrieved from https://developer.android.com/about/versions/marshmallow/android-6.0-changes
[15] Pardeshi, A. (2025). Early stage voice phishing detection with runtime permission request (Project Stage-I Report). Deogiri Institute of Engineering and Management Studies, Dr. Babasaheb Ambedkar Technological University, Lonere-Raigad.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY