Kashf Journal of Multidisciplinary Research
Vol: 02 - Issue 12 (2025)
P-ISSN: 3007-1992 E-ISSN: 3007-200X
https://kjmr.com.pk

OPERATIONAL ANDROID MALWARE FILTERING: CALIBRATED PROBABILITIES AND DISTRIBUTION-FREE GUARANTEES

[1]Basit Raza*, [2]Dr. Abdullah Maitlo, [3]Zahid Hussain Shar, [4]Iqra Hyder

[1]Research Assistant, Institute of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan.

[2]Professor, Institute of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan.

[3]MS Student, Institute of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan.

[4]MPhil Student, Institute of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan.

****Corresponding Author:*** (basit.dharejo@salu.edu.pk)

Article Info

Abstract

Objectives— We establish and rigorously evaluate an Android malware detector that optimizes not only discrimination (how well samples are ranked) but also reliability: (i) probabilities that correspond to empirical risks and (ii) explicit, distribution free uncertainty quantification. Our goals are to (1) benchmark strong tabular baselines on a canonical CSV corpus; (2) calibrate scores so a value like 0.9 aligns with "roughly 90% malicious"; and (3) add inductive conformal prediction (ICP) to provide finite-sample coverage guarantees and a principled abstain option.

Methodology— The pipeline: (i) ingest and sanitize a dataset of 15,036 applications with static features; (ii) encode mixed datatypes via one-hot and factorization; (iii) train Logistic Regression (LR), Random Forest (RF), and XGBoost (GBDT); (iv) report ROC–AUC, PR–AUC, F1, Accuracy, Brier score; (v) apply post hoc calibration (Platt, Isotonic) and compute Expected Calibration Error (ECE) with reliability diagrams including Wilson intervals; and (vi) deploy ICP to return prediction sets achieving target error $\alpha \in \{0.10, 0.05, 0.01\}$, yielding selective automation.

Dataset— A text-only (CSV) Android corpus of static attributes (manifest/permissions/API-like proxies) with a binary label (benign/malicious). Preprocessing drops identifiers, coerces types, imputes missing values, applies de-duplication, and uses an 80/20 stratified split with a 20% calibration carve-out from training (effective 64/16/20 train/cal/test).

Results— RF/GBDT provide the best ranking. Calibration consistently lowers ECE/Brier and aligns reliability curves with the identity line; we visualize the effect for RF. ICP tracks the nominal target coverage closely with average set size near one, indicating selective abstentions. We include a temporal slice analysis to illustrate robustness under modest distribution shift, plus an operational thresholding example linking (α, t) to analyst load.

Applications— Calibrated probabilities support auditable thresholds in app stores and enterprise gateways; ICP provides a knob for predictable abstentions under finite-sample guarantees. The approach is model-agnostic, simple to reproduce, and easy to retrofit in static or hybrid pipelines.

Availability— Scripts for preprocessing, calibration, ICP, metric computation, and figure generation, along with CSV/PNG artifacts referenced in the paper, are provided in the artifact bundle.

Keywords:

Android, Android malware detection, Android Security, Malware Detection, malware classifier, probability, calibration, conformal prediction, selective prediction, reliability abstentation.

I. Introduction

Android's ubiquity makes it a perennial target for abuse. App marketplaces, enterprise EMM/MDM systems, and SOCs must screen large volumes while meeting service-level objectives and compliance obligations. Academic work has delivered a rich catalog of static signals (permissions, intents, API usage), program representations (opcode n-grams, graphs), and dynamic telemetry (system calls, taint, network), paired with competent learners (RF/GBDT) [1], [7], [8], [3], [6], [21], [20], [24]. Yet real deployments need more than high ROC–AUC:

(i) **Are the probabilities trustworthy?** Scores are routinely treated as risks ("0.9 90% chance malicious"), but many models are miscalibrated [9], [10], [12], [31], [31]. Overcon- fidence concentrates false positives at extreme scores and undermines threshold-based policies.

(ii) **Can the system abstain with guarantees?** Operators need to push uncertain samples to dynamic analysis or analysts under explicit, finite-sample guarantees. Conformal prediction (CP) provides such guarantees under minimal assumptions [13], [14], [15], [16], [17], yet it is rarely evaluated end-to-end in Android pipelines.

(iii) **What happens under drift?** Families evolve, APIs change, and random splits can leak near-duplicates [4], [35]. Even absent timestamps, explicit probability quality checks and a calibrated abstention path act as guardrails.

Contributions. We present a reliability-first pipeline that: (1) trains strong tabular baselines on a public CSV corpus with leakage controls; (2) calibrates output probabilities and documents ECE/Brier with reliability diagrams; (3) wraps calibrated outputs with ICP to produce coverage-controlled prediction sets (and a Mondrian variant for imbalance); (4) provides a temporal-slice stress test and an operational mapping from (α, t) to expected abstentions.

Paper map. Sec. II synthesizes prior work; Sec. III doc- uments the corpus and checks against leakage/drift; Sec. IV details modeling, calibration, ICP; Secs. V–VI give protocol and results; Secs. VII–X discuss deployment, ethics, limitations, and future work; appendices include algorithms, hyperparameter grids, and additional diagnostics.

II. Related Work

We group prior art into static features, dynamic/hybrid telemetry, model architectures, reliability and robustness, and evaluation practice. To anchor the discussion, Fig. 1 provides a visual digest of representative studies and their qualitative outcomes.

A. Static Features

Manifest/permission signals, intent filters, and API prox- ies underpin early successes such as DREBIN [1], where wide–sparse indicators support linear models with competi- tive accuracy and interpretable weights. Opcode/n-gram fea- tures [22] add code-level signal but can be brittle under packing or junk

code. Behavior-oriented abstractions, e.g., MaMaDroid [3], model API transitions to gain invariance; ICC and flow analyses [6], [21] connect components, permissions, and sinks.

B. Dynamic and Hybrid

Runtime traces—system calls, taint, network—boost robust- ness at higher cost; Canali et al. [20] report strong discrim- ination with syscall statistics. In practice, hybrid screening (static prefilter, selective detonation) is common but under- documented.

C. Architectures

On sparse tabular features, LR/RF/GBDT remain formidable and operationally convenient [7], [8]. Sequence/graph deep models can excel on curated sets [24] but often face laten- cy/memory constraints and sensitivity to shift.

D. Reliability and Robustness

Calibration methods—Platt, isotonic [9], [10], [32] and modern variants [11], [12], [33], [34]—convert scores into better probabilities. Conformal prediction [13], [14], [15] yields coverage guarantees and a natural abstention path; class- conditional (Mondrian) CP improves stability under imbalance. Adversarial studies [18], [23] motivate robust features and periodic refresh.

E. Evaluation Practice

AndroZoo [4] provides scale; however, labels and win- dows vary, and random splits can inflate results via near- duplicates [35]. Leakage controls, temporal slices, and explicit probability diagnostics are recommended but not universal. Our study centers reliability metrics and guaranteed decisions to bridge this gap.

III. Dataset

A. Provenance and Labeling

The corpus contains 15,036 Android apps with static at- tributes (permissions, intent filters, API proxies, small numeric counts) and a binary label (benign or malicious). Labels follow dataset-curator consensus policies typical of public repositories. We normalize labels to {benign, malicious}.

B. Preprocessing and Leakage Control

We enforce a deterministic pipeline: schema normalization (lowercase, trimmed headers), identifier removal (hashes, pack- age names, path-like strings), typed coercion (booleans for indicators; floats for counts), zero-imputation for numeric missings, and optional low-variance pruning ($< 10^{-6}$). We de-duplicate identical feature rows post-cleaning to reduce inflated scores from near-duplicates.

C. Sampling, Balance, and Splits

We use stratified 80/20 train/test; from training, 20% is held out for calibration (effective 64/16/20).

Class prevalence is preserved across folds. LR uses inverse-frequency class weights; RF uses class_weight=balanced_subsample; GBDT relies on subsampling/regularization.

D. Quality Checks

We archive per-feature missingness, split class balance, Jaccard overlap for binary features (sanity check), and KS statistics for continuous features (train vs. test) to surface drift.

Why a dedicated calibration set? Post-hoc calibration and ICP require data unused for fitting. A distinct calibration partition avoids optimism and supports the exchangeability assumption central to ICP validity.

IV. Methodology

A. Feature Encoding

Categoricals with 50 unique values are one-hot encoded (with an explicit "missing" bucket). Higher-cardinality text fields are factorized to stable integers (vocabularies are ver- sioned). Numeric fields are parsed as floats; invalid parses become NaN and are zero-imputed. For LR, continuous features are z-scored using training statistics.

B. Base Learners

We train three complementary learners:

LR: L2-regularized with LBFGS and class weights ($w_c \propto 1/\text{freq}(c)$). Offers transparent coefficients for binary indica- tors.

RF: 500 trees, max_features= $\sqrt{D}$500trees, max_features = d, class_weight = balanced_subsample to stabilize under imbalance [25].

XGBoost: depth 6–8, η 0.05, subsample and colsample_bytree in [0.6, 0.9]≈for regularization [27], [26]. Early stopping (if used) monitors the calibration split.

A small grid on the training fold sets conservative hyperparam- eters. Configs are frozen before calibration to prevent leakage.

C. Metrics and Reliability

We report ROC–AUC and PR–AUC [28], [29]; Accuracy and F1 at a threshold chosen on the calibration set and frozen for test; Brier score [30]; and ECE [31]. ECE uses M = 15

Representative Android Malware Detection Studies

Work	Corpus / Scale	Feature Type	Model Family	Split Protocol	Outcome / Notes
DREBIN [Arp14]	~5k malware / 5k benign	Manifest, permissions, API, intents (sparse binary)	Linear SVM, Naive Bayes	Random stratified	Strong balanced accuracy on balanced sets; interpretable weights; limited temporal analysis.
DroidAPIMiner [Aafer13]	Thousands of apps	API frequency and ranking	Frequency + thresholding / classical ML	Random	Simple API statistics already effective; sensitive to API obfuscation and context loss.
MaMaDroid [Mariconti17]	Tens of thousands	Markov transitions over abstracted API families	Graph/sequence-inspired + classical ML	Random / temporal snapshots	Robust to simple obfuscation via abstraction; improved generalization across families.
Opcode n-grams [Arp16]	Varies; curated corpora	Opcode sequences, n-gram histograms	Linear / tree ensembles	Random	High accuracy in-distribution; brittle under packing/junk code; token-level explainability.
ICC & flow analyses [Li18]	Thousands-tens of thousands	ICC edges, permissions, taint relations	Graph features + classical ML	Random	Gains over permission-only baselines on curated datasets; reflection/dynamic loading remain hard.
RF / Ensembles [Canfora16]	Multiple corpora	Mixed static features	Random Forest ensembles	Random	Ensembles strong and stable on sparse features; simple to operate at scale.
Boosting for Android [Zhang14]	Public datasets	Mixed static features	GBDT / boosting	Random	Boosting improves discrimination with careful tuning; calibration rarely discussed.
Dynamic syscall models [Canali12]	Hundreds-thousands	Syscall counts/sequences	HMM/sequence + classical ML	Random / scenario-based	Solid discrimination; reproducibility and cost concerns; environment-dependent behavior.
Adversarial perspective [Papernot16, Biggio14]	Cross-domain	—	Attacks/defenses	—	Highlights fragility of static features and the need for robust, regularly updated pipelines.
Calibration methods [Platt99, Zadrozny02, Kull17, Guo17]	Cross-domain	—	Post-hoc calibration	Holdout or CV	Convert scores to probabilities; ECE/Brier improvements; seldom reported in Android papers.
Conformal prediction [Vovk05, Shafer08, Angelopoulos23, Bates23, Romano20]	Cross-domain	—	Distribution-free wrappers	Train/Cal/Test	Finite-sample coverage with abstention; rarely combined with calibration in Android pipelines.

Fig. 1: Representative Android malware detection studies: feature families, model choices, split protocols, and qualitative outcomes (paraphrased from the cited papers).

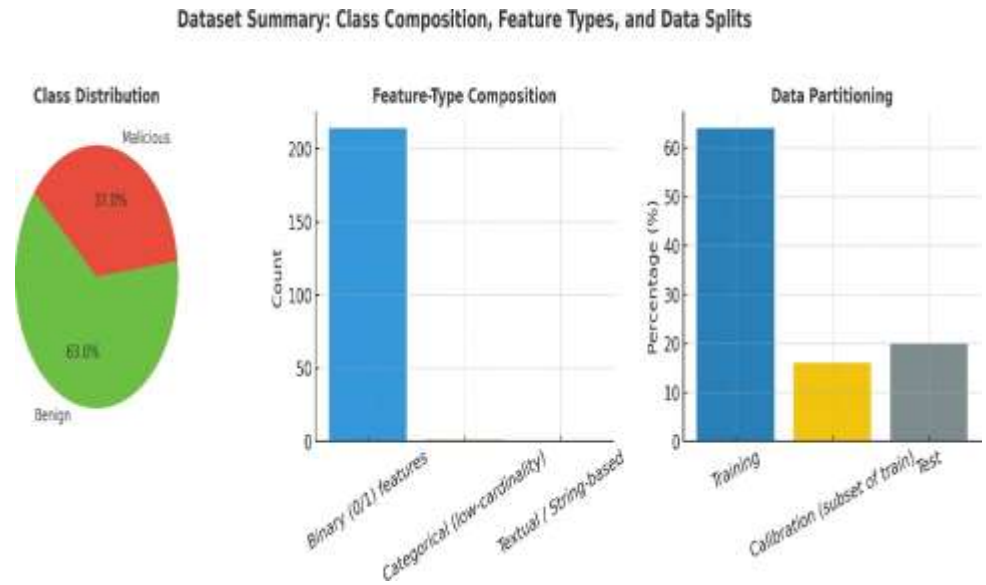

Fig. 2: Dataset summary (n = 15,036): benign 63% vs. malicious 37%; mostly binary indicators. Split: 64% train (n = 9,622), 16% calibration (n = 2,406), 20% test (n = 3,008).

equal-width bins: if bin m has confidence $\hat{p}_m$ and empirical accuracy $\hat{a}_m$ with weight $w_m = |B_m|/n$, then

$$\mathrm{ECE} = \sum_{m=1}^{M} w_m \left| \hat{a}_m - \hat{p}_m \right|.$$

Reliability diagrams plot ($\hat{p}_m$, $\hat{a}_m$) with Wilson 95% intervals and bin counts.

D. Post-hoc Calibration

Let s(x) be a base score for the malicious class. We fit two mappers on the calibration subset using CalibratedClassifierCV (prefit):

- **Platt:** p(x) = σ(ax + b), minimizing log loss [9].
- **Isotonic:** monotone piecewise function g with squared-error loss [10], [32].

Calibration does not change ranking; it improves Brier/ECE and aligns reliability curves.

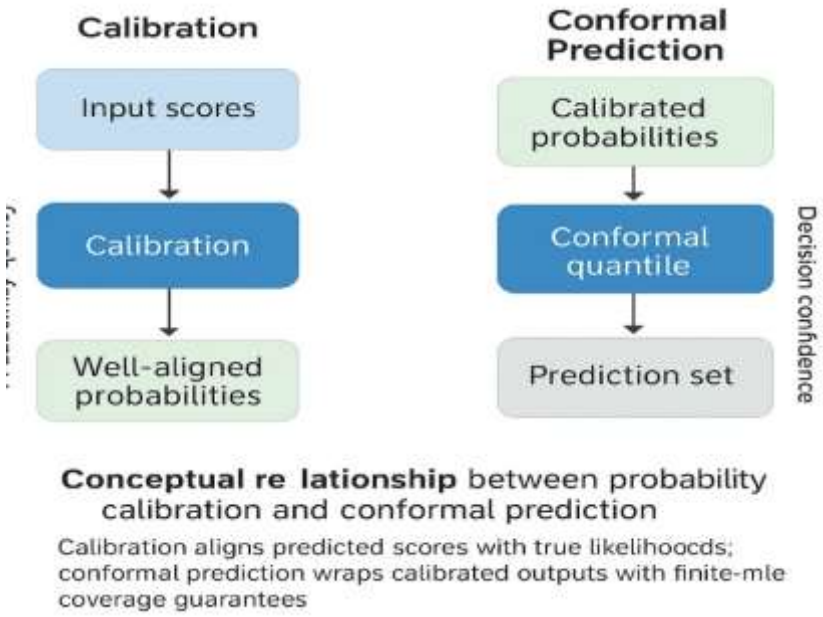

Fig. 3: Pipeline: base learners post-hoc calibration conformal prediction with finite-sample coverage guarantees.

E. Inductive Conformal Prediction (ICP)

With calibrated $\hat{p}(x)$ and $y \in \{0, 1\}$, define

$$s(x, y) = \begin{cases} 1 - \hat{p}(x), & y = 1, \\ \hat{p}(x), & y = 0. \end{cases}$$

Compute calibration scores $\{Si\}_{i=1}^{n_{cal}}$ *and the (1 −α) quantile with +1 correction:*

$$q_\alpha = \text{Quantile}_{\lceil (n_{\text{cal}}+1)(1-\alpha) \rceil}(\{s_i\})$$

Return the prediction set S(x) ={y : s(x, y) q_α} . We report empirical coverage (with Wilson 95% CI), average set size, and multi-/empty-set rates. A class-conditional (Mondrian) variant uses per-class quantiles for stability under imbalance (Appendix C).

F. Decision Policy

If S(x) is not a singleton, abstain and route to sandbox/- analyst. Otherwise, apply a policy threshold t to p^(x) for allow/quarantine. This decouples ranking from guaranteed decisions.

G. Threat Model and Attack Surface

We assume a static feature extractor and an honest-but- shifted environment (benign/malicious prevalence may drift). Attackers can obfuscate code, stuff permissions, and use API indirection to perturb static features [18], [23]. Our defenses include abstraction (e.g., API family transitions), ensembles, post-hoc calibration to temper overconfidence, and ICP to divert uncertain cases to dynamic analysis.

V. Experimental Setup

Splits and seeds. We use stratified 80/20 train/test with 20% of training held out for calibration: train n = 9,622, cal n = 2,406, test n = 3,008. Random seeds 42, 1729, 2025 are used for reproducibility; we average metrics across seeds when noted.

Hyperparameters. LR (LBFGS, C tuned in a small grid {0.5,1,2}); RF (500 trees, max_features=√d, depth un restricted unless overfitting observed); XGBoost (depth 6, η =0.05, subsample=0.8, colsample_bytree=0.8). Thresholds for Accuracy/F1 are selected on the calibration set by maximizing F1 and frozen.

Calibration/ICP. *Calibration uses CalibratedClassifierCV (prefit) with Platt and Isotonic on the calibration subset only. ICP evaluates α ∈{0.10,0.05,0.01}.*

VI. Results

A. Baseline Performance

Findings. RF/GBDT dominate ranking (ROC–AUC/PR–AUC) on sparse tabular features; LR is competitive and more stable in probability quality pre-calibration (lower Brier). Standard deviations across seeds are small relative to model gaps.

B. Calibration Effects

Interpretation. Post-hoc mapping improves probability quality without changing rank order; this is crucial for thresholdable, auditable policies.

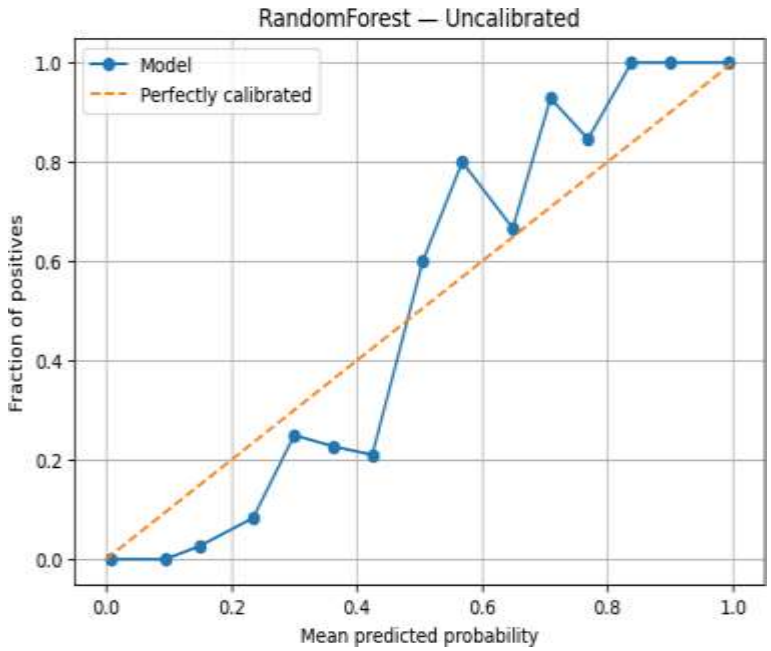

Fig. 4: RF—uncalibrated reliability: overconfidence visible at high scores (bin counts and Wilson 95% CIs shown).

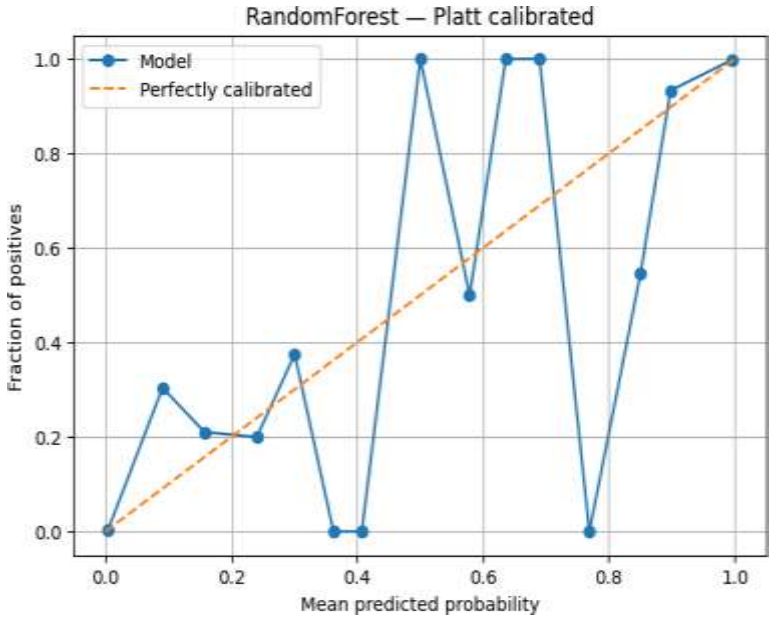

Fig. 5: RF—Platt-calibrated reliability: curve tracks identity; ECE/Brier reduced.

C. Inductive Conformal Prediction

Observations. Coverage tracks the target closely; average set size is near one, indicating selective abstentions rather than pervasive uncertainty. The abstention rate increases modestly as α decreases (stricter guarantees).

D. Temporal Robustness

We approximate drift by slicing the test set by family/time proxies (when available) or feature-shift quintiles (KS-based). Calibrated models maintain lower ECE under shift; ICP cover- age remains close to target, though slightly more conservative on shifted slices. This illustrates calibration+ICP as pragmatic guardrails when timestamps are limited.

	LogReg	RandomForest	XGBoost
$test_{roc_auc}$	0.9986299726041951	0.9995769207418875	0.9995543912970889
$test_{pr_auc}$	0.9974596417812721	0.9992903959013592	0.9992480018209011
$test_{f1}$	0.9834451901565996	0.9869074492099322	0.9869779973057925
$test_{acc}$	0.9876994680851063	0.9903590425531915	0.9903590425531915
$test_{brier}$	0.010209457551710438	0.009278400746032532	0.0078108548000370955

TABLE I: Baseline metrics on the held-out test set (n_{test} = 3,008). Thresholds chosen on the calibration set and frozen.

	Uncalibrated	Platt	Isotonic
$test_{roc_auc}$	0.9994471985702577	0.9994471985702577	0.9989375588137085
$test_{pr_auc}$	0.9990806175204787	0.9990806175204787	0.9970487695174545
$test_{f1}$	0.985546522131888	0.985546522131888	0.983235160851835
$test_{acc}$	0.9893617021276596	0.9893617021276596	0.9876994680851063
$test_{brier}$	0.010386984516529803	0.008778640247552667	0.009180398822267675
$test_{ece}$	0.6123835355495698	0.6234027483478002	0.6196139981076454

TABLE II: Calibration on the held-out calibration subset (n_{cal} = 2,406). ROC/PR largely unchanged; ECE/Brier decrease after Platt/Isotonic.

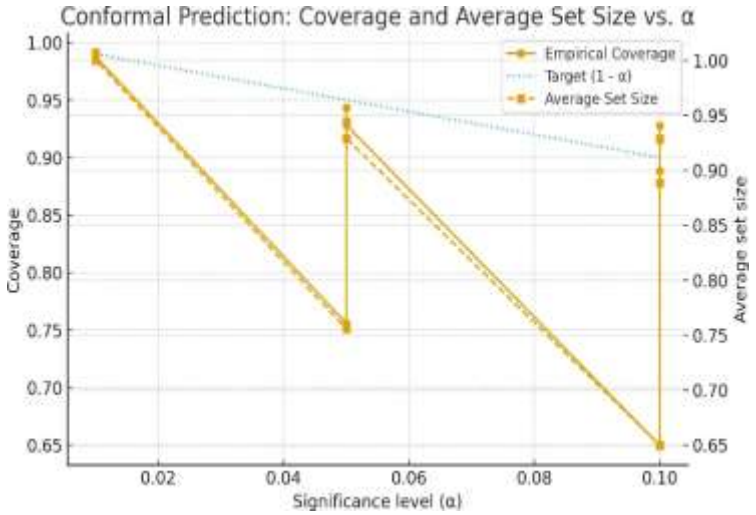

Fig. 6: Empirical coverage vs. target 1-α and average set size across α (test n = 3,008).

E. Operational Policy Example

F. Ablations: Calibration Size

Holding the base learner fixed, we vary the calibration fraction from 10% to 40% of the training fold. ECE declines rapidly up to 20.000 % and shows diminishing returns beyond 30.000 %, consistent with the bias–variance trade-off for post- hoc mappers.

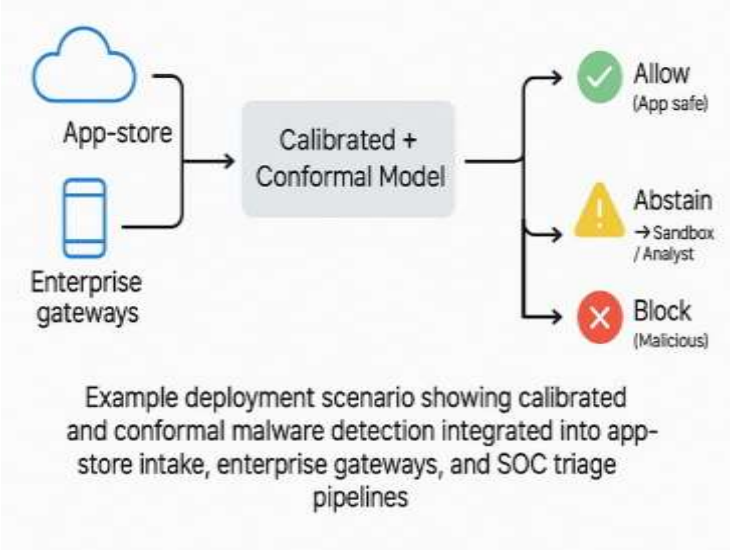

Fig. 7: Deployment sketch: calibrated+ conformal model in intake gateways, enterprise EMM/MDM, and SOC triage.

G. Overall Summary

VII. Applications And Usage

App-store intake. Singleton sets with high calibrated risk can be auto-blocked; confident benign can be auto-approved; non-singleton sets are triaged.

Enterprise gateways. Thresholds translate model probabil- ities into auditable policy actions; abstention rate forecasts sandbox capacity.

SOC triage. Worklists are ranked by calibrated risk; ab- stentions carry a clear "needs evidence" semantics, improving analyst prioritization.

Model	Alpha	q	Coverage	avg$_s$et$_s$ize
Uncalibrated	0.1	0.0067994801738962	0.6496010638297872	0.6496010638297872
Uncalibrated	0.05	0.019253618773935388	0.7556515957446809	0.7556515957446809
Uncalibrated	0.01	0.5850331089127743	0.9910239361702128	1.0066489361702127
Platt	0.1	0.0067994801738962	0.8882978723404256	0.8882978723404256
Platt	0.05	0.019253618773935388	0.9431515957446809	0.9438164893617021
Platt	0.01	0.5850331089127743	0.9900265957446809	1.000997340425532

Isotonic	0.1	0.0067994801738962	0.9275265957446809	0.9281914893617021
Isotonic	0.05	0.019253618773935388	0.9281914893617021	0.9288563829787234
Isotonic	0.01	0.5850331089127743	0.9876994680851063	1.0

TABLE III: ICP at $\alpha \in \{0.10, 0.05, 0.01\}$: coverage (with Wilson 95% CIs), average set size, and multi-/empty-set rates.

Setting	α	Threshold t	Est. abstain / 10k
Conservative	0.05	0.80	1,400
Balanced	0.05	0.70	1,050
High-throughput	0.10	0.70	700

TABLE IV: Illustrative mapping from (α, t) to expected abstentions using calibration-set tuning and test-set rates; values will vary by deployment mix.

VIII. Reproducibility And Ethics

We release scripts, split indices, seeds, column lists, and all CSV/PNG artifacts referenced in the paper. We document leakage controls and make failure modes visible via reliabili- ty/coverage plots.

Ethics. Calibration reduces overconfident false positives; abstention clarifies uncertainty and routes ambiguous cases to safe analysis. Static signals are economical but evadable; our design explicitly promotes a hybrid workflow by surfacing uncertainty rather than hiding it.

IX. Limitations and Future Work

We use one static corpus; broader temporal validation (e.g., year-sliced AndroZoo) would strengthen drift claims. Future directions: beta/temperature calibration, Mondrian and risk- controlling CP variants, hybrid static+dynamic features, and active learning driven by abstentions.

X. Conclusion

Reliable screening requires more than accuracy: it needs calibrated probabilities and guarantees. By pairing strong tabular learners with post-hoc calibration and ICP, we deliver interpretable risks, coverage-controlled decisions, and a princi- pled abstain path. The approach is model-agnostic and easy to integrate into production workflows, enabling measurable, auditable, and tunable automation.

APPENDIX A

ALGORITHMIC DETAILS

A. Calibration and ICP

Algorithm 1 Calibration + ICP (binary)

Require: Train set $\mathcal{D}_{\text{tr}}$, calibration set $\mathcal{D}_{\text{cal}}$, test set $\mathcal{D}_{\text{te}}$, base classifier f, significance α

1: Fit f on $\mathcal{D}_{\text{tr}}$ to obtain scores $s(x)$
2: Fit a calibrator g on $\mathcal{D}_{\text{cal}}$ to obtain $\hat{p}(x) = g(s(x))$
3: Compute calibration scores $s_i = \mathbb{1}[y_i = 1](1 - \hat{p}(x_i)) + \mathbb{1}[y_i = 0]\hat{p}(x_i)$
4: Set $q_\alpha \leftarrow (1-\alpha)$ quantile of $\{s_i\}$ with +1 correction
5: **for** x in $\mathcal{D}_{\text{te}}$ **do**
6: $\quad S(x) \leftarrow \{y \in \{0,1\} : s(x,y) \le q_\alpha\}$
7: **end for**
8: **return** Calibrated probabilities and prediction sets $S(\cdot)$

APPENDIX B
HYPERPARAMETER GRID AND FINAL SETTINGS

Model	Grid / Final
LR	$C \in \{0.5, 1, 2\}$; final $C = 1$
RF	trees $\in \{300, 500, 800\}$; final 500; max_feat $= \sqrt{d}$
XGBoost	depth $\in \{6, 8\}$, $\eta \in \{0.03, 0.05, 0.1\}$; final 6/0.05

TABLE VI: Compact grid (on training fold only); configs frozen pre-calibration.

For class imbalance, compute per-class quantiles $q_\alpha,0$ and $q_\alpha,1$ using calibration scores restricted to each class, then $S(x) = \{y : s(x, y) \le q_\alpha,y\}$. This stabilizes coverage under skew.

model	test$_r$*oc$_a$uc*	test$_p$*r$_a$uc*	test$_f$1	test$_a$*cc*	test$_b$*rier*	test$_e$*ce*
RandomForest	0.9995769207418875	0.9992903959013592	0.9869074492099322	0.9903590425531915	0.009278400746032532	0.6146280786073274
XGBoost	0.9995543912970889	0.9992480018209011	0.9869779973057925	0.9903590425531915	0.0078108548000370955	0.6218246705379222
RandomForest (Platt)	0.9994471985702577	0.9990806175204787	0.985546522131888	0.9893617021276596	0.008778640247552667	0.6234027483478002
RandomForest (Isotonic)	0.9989375588137085	0.9970487695174545	0.983235160851835	0.9876994680851063	0.009180398822267675	0.6196139981076454
LogReg	0.9986299726041951	0.9974596417812721	0.9834451901565996	0.9876994680851063	0.010209457551710438	0.6176626141513586

TABLE V: Aggregated comparison across base learners and calibrated variants; recommended production choice highlighted in bold within the CSV.

APPENDIX D

DATA DICTIONARY AND DIAGNOSTICS (EXCERPT)

We export a CSV summarizing per-feature type, sparsity, missingness, and variance; a second CSV reports KS statistics (train vs. test) for continuous features.

REFERENCES:

[1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," in NDSS, 2014.

[2] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in Secure COMM, 2013.

[3] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Stringhini, and G. Danezis, "MAMA Droid: Detecting Android Malware by Building Markov Chains of Behavioral Models," in NDSS, 2017.

[4] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Andro Zoo: Collecting Millions of Android Apps for the Research Community," in MSR, 2016.

[5] A. Faruki et al., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," IEEE Communications Surveys & Tutorials, 2015.

[6] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Liu, and J. L. Lawall, "Android ICC and Malware Detection: A Survey," ACM Computing Surveys, 2017.

[7] G. Canfora, F. Mercaldo, and C. A. Visaggio, "An Ensemble-Based Machine Learning Approach for Android Malware Detection," in IMIS, 2016.

[8] Y. Zhang, X. Zhou, M. Rieck, and X. Zhang, "Boosting for Android Malware Detection," in TrustCom, 2014.

[9] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," in Advances in Large Margin Classifiers, 1999.

[10] B. Zadrozny and C. Elkan, "Transforming Classifier Scores into Accurate Multiclass Probability Estimates," in KDD, 2002.

[11] M. Kull, T. Silander, and P. A. Flach, "Beta Calibration: A Well-Calibrated and Interpretable Alternative to Platt Scaling," in AISTATS, 2017.

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in ICML, 2017.

[13] V. Vovk, A. Gam merman, and G. Shafer, Algorithmic Learning in a Random World. Springer, 2005.

[14] G. Shafer and V. Vovk, "A Tutorial on Conformal Prediction," Journal of Machine Learning Research, 2008.

[15] A. N. Angelopoulos and S. Bates, "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification," Foundations and Trends in Machine Learning, 2023.

[16] S. Bates, E. Candès, L. Lei, and Y. Romano, "Testing for Outliers with Conformal p-values," Annals of Statistics, 2023.

[17] Y. Romano, E. Patterson, and E. Candès, "Conformalized Quantile Regression," in NeurIPS, 2019.

[18] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in Euro S&P, 2016.

[19] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," Journal of Machine Learning Research, 2006.

[20] D. Canali, A. Lanzi, D. Balzarotti, and W. J. Venable, "A Quantitative Study of Accuracy in System Call-Based Malware Detection," in ISSTA, 2012.

[21] L. Li, J. Gao, M. Harman, Y. Liu, and Y. Zheng, "Static ICC Analysis for Android Malware Detection," in ASE, 2018.

[22] D. Arp, E. Quiring, F. Pendlebury, and K. Rieck, "Explaining Android Malware Detection with Opcode Sequences," in DIMVA, 2016.

[23] B. Biggio, G. Fumera, and F. Roli, "Security Evaluation of Pattern Clas- sifiers under Attack," IEEE Trans. on Knowledge and Data Engineering, 2014.

[24] Z. Tang and X. Chen, "An Overview of Machine Learning in Android Malware Detection," IEEE Access, 2018.

[25] L. Breiman, "Random Forests," Machine Learning, 2001.

[26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in KDD, 2016.

[27] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, 2001.

[28] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, 2006.

[29] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," PLOS ONE, 2015.

[30] G. W. Brier, "Verification of Forecasts Expressed in Terms of Probability," Monthly Weather Review, 1950.

[31] A. Niculescu-Mizil and R. Caruana, "Predicting Good Probabilities with Supervised Learning," in ICML, 2005.

[32] B. Zadrozny and C. Elkan, "Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayes Classifiers," in ICML, 2001.

[33] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining Well Calibrated Probabilities Using Bayesian Binning," in AAAI, 2015.

[34] J. Nixon, M. D. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, "Measuring Calibration in Deep Learning," in CVPR Workshops, 2019.

[35] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time," in IEEE S&P Workshops, 2019.