Vol. 8, No. 3, June 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

PADLOCK: A Context-Aware On-Device System for Android Permission-Risk Assessment – A Tanzanian Case Study

Ansgar A. Mtunga[1], Bonny Mgawe[2], Judith Leo[3]

[1,2,3]School of Computational and Communication Science and Engineering, Nelson Mandela African Institution of Science and Technology, Tanzania.

Received:
September 29, 2025
Revised:
March 30, 2026
Accepted:
May 30, 2026
Published:
July 26, 2026

Corresponding Author:
Author Name*:
Ansgar A. Mtunga
Email*:
mtungaa@nm-aist.ac.tz

DOI:
10.63158/journalisi.v8i3.1586

Abstract. Android applications often request sensitive permissions beyond their functional needs, creating privacy and security risks, especially in financial and digital lending apps. Existing Android permission mechanisms provide limited visibility into how permissions are used at runtime. This study proposes PADLOCK, a context-aware prototype for event-driven permission monitoring and contextual risk assessment. The methodology combined a survey-based user study (n = 600), analysis of 3,816 applications, machine learning development, and Android prototype implementation. A compact deep neural network, ShieldAI, was trained on permission-based features with heuristic risk labels to classify applications as benign or potentially higher-risk. The model was integrated into PADLOCK to support on-device monitoring and user-guided permission decisions. On a held-out test set, ShieldAI achieved 96.73% accuracy, 94.82% precision, and 92.42% recall. Controlled on-device testing showed an inference latency of approximately 2–3 ms per prediction and indicated reduced higher-risk permission-granting behaviour in PADLOCK-assisted evaluations. This study contributes an integrated Android prototype combining contextual permission analysis, machine-learning-based risk assessment, and event-driven monitoring. However, the findings remain limited to the evaluated dataset and controlled conditions, and broader real-world validation is required.

Keywords: Android security; permission-risk assessment; machine learning, on-device inference, mobile privacy, financial applications.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1 INTRODUCTION.

Smartphones have become integral to communication, healthcare, business, education, and financial services. The rapid growth of mobile applications has expanded the Android ecosystem to millions of applications distributed through platforms such as the Google Play Store [1]. Many of these applications access user-related data and device resources, including contacts, location and media files, raising significant security and privacy concerns. The Android operating system regulates such access through a permission-based security mechanism [2]. To make users more aware, newer versions of Android have included more detailed restrictions, such as runtime permissions (Android 6.0+), one-time permissions (Android 10), automated revocation of unnecessary rights (Android 11), and the Privacy Dashboard (Android 12+). However, once users grant permissions, they often remain active until manually revoked [3]. In practice, however, the platform does not distinguish between legitimate and potentially risky uses of the same data. For example, Hayes et al. [4] observed that six popular applications collected substantial amounts of data beyond what was necessary for their declared permissions. Mishra B et al. [5] reported that permission abuse is common, with some applications excessively seeking and utilising permissions for commercial objectives, resulting in the collection of user data for advertising or profiling without user awareness. These findings indicate that existing permission-control mechanisms cannot distinguish legitimate use from abuse. This issue reflects a key limitation of Android's permission model, which provides limited consideration of contextual application behaviour.

Globally, the rapid growth of mobile applications, particularly within financial and digital lending services, has increased concerns regarding excessive permission requests and the misuse of sensitive user data. In Sub-Saharan Africa, the challenge is especially pronounced because of the growing adoption of mobile financial services and digital lending platforms. These applications frequently process sensitive information, including personal identities, financial records, contacts, and location data, increasing users' exposure to privacy and security risks [6]. Recent studies have also reported widespread permission misuse, including coercive debt-collection practices such as harassment and public shaming of borrowers. In East Africa, analyses of Android digital lending applications have further revealed extensive access to contacts, SMS messages, location,

and media data, together with evidence that user information may be transmitted before meaningful consent is obtained [7], [8].

In Tanzania, the rapid expansion of mobile financial services has further intensified these privacy and security concerns. Tanzania counted over 76 million mobile-money accounts across platforms such as M-Pesa, TigoPesa, Airtel Money, and HaloPesa, driving significant growth in digital lending services [9]. Yet this expansion has also exposed serious privacy risks. Regulators such as the Tanzania Communications Regulatory Authority (TCRA) and the Bank of Tanzania (BOT) uncovered unlicensed lenders scraping borrowers' contacts and social media accounts to publicly shame debtors. Between 2022 and 2024, investigations further revealed predatory lenders misusing Android permissions to directly harass borrowers [10]. As a result, 69 Android applications were removed from distribution within the country [11].

These challenges have motivated extensive research into Android permission security and privacy protection. Almomani et al. [12] analysed permission use patterns over many years and across several categories, comparing benign and higher-risk applications over time. For instance, W. Wang et al. [13] proposed a hybrid static/dynamic technique that combines machine learning analysis of permission request patterns with runtime memory graph analysis for application classification. Other studies, such as Ayres-Pereira [14], examine user perceptions of privacy and compliance with privacy requirements. Regulatory frameworks like the GDPR require applications to disclose all collected data [15], but practical assessments show that this is not happening in many cases. Similarly, Android's security approach does not provide mechanisms for auditing applications during runtime. Once permission is given, there is no way to track when and how an app actually uses granted permissions. This reveals a critical gap in current Android security mechanisms: while permissions regulate access, they do not provide context-aware monitoring of permission usage during application operation. Most existing approaches primarily rely on static analysis or offline detection methods, which may be insufficient for capturing changes in permission usage patterns and application behaviour over time. To address this limitation, this study proposes PADLOCK, a context-aware, on-device Android security prototype designed to monitor permission activity, analyse contextual permission usage patterns, and support user-guided permission management. Unlike existing approaches that primarily rely on static analysis or offline detection, PADLOCK

integrates lightweight machine-learning-based permission-risk assessment, event-driven monitoring, and contextual analysis within a unified on-device architecture to support more informed privacy risk assessment during application use.

The contribution of this study is threefold: (i) an empirical analysis of permission-related privacy risks among Android users, (ii) the development of a lightweight machine learning model for permission-based risk classification, and (iii) the design and implementation of an on-device prototype system for event-driven permission monitoring and contextual risk assessment. The study is evaluated within the context of financial and digital lending applications, which prior studies have identified as exhibiting elevated privacy and security sensitivity [16], [17], while remaining applicable across broader Android application categories.

2 MATERIALS AND METHODS

This study employed a multi-phase methodology to investigate permission-related privacy risks in Android applications and to design, implement, and evaluate the PADLOCK system. The methodology integrates user behavioural evidence, machine-learning model development, and system-level experimental validation to support the evaluation of the proposed approach. This study adopted a multi-phase research design combining quantitative experimentation, survey-based behavioural assessment, and design science.

2.1 Research Methodological Framework

The methodological framework consisted of three interconnected stages. First, a survey-based study examined users' awareness, behaviour, and experiences related to Android permissions to inform system requirements. Second, the ShieldAI machine-learning model was developed through dataset preparation, heuristic labelling, feature engineering, model training, and evaluation. Finally, the PADLOCK prototype was implemented using a design science approach, integrating event-driven monitoring, contextual analysis, and on-device inference within an Android environment [18].

2.2 Study Area and Case Study Justification

This study was conducted in Tanzania, which served as the case study for investigating Android permission misuse and evaluating the PADLOCK prototype. Tanzania provides a

suitable setting due to the rapid growth in smartphone adoption, with approximately 26.9 million smartphone users reported by the Tanzania Communications Regulatory Authority (TCRA) as of September 2025 [9]. Recent regulatory audits have also identified excessive permission requests and personal data over-collection among higher-risk mobile applications [11], highlighting the need for improved permission-risk assessment. Survey participants were recruited primarily from Dar es Salaam, Arusha, and Dodoma to capture diverse smartphone user experiences. Although specific application domains are referenced to illustrate permission-risk patterns, the study maintains a general focus on Android permission misuse across multiple application categories.

2.3 Survey Design and Participants

The study targeted Android smartphone users who regularly install and use mobile applications requiring runtime permissions. Participants were recruited primarily from Dar es Salaam, Arusha, and Dodoma using purposive and convenience sampling. Purposive sampling was used to select Android smartphone users who regularly install and use mobile applications, while convenience sampling facilitated recruitment of eligible participants through online (Google Forms) and face-to-face survey administration.
The minimum survey sample size was estimated using Yamane's formula [19]:

$$n = \frac{N}{1+N(e^2)} \quad (1)$$

where:

n represents the required sample size,
N is the target population,
e is the level of precision (acceptable sampling error 5%).

Although the minimum recommended sample was approximately 400 participants, the study targeted 600 respondents to improve statistical reliability and provide a margin for potential incomplete responses. Data were collected through both online (Google Forms) and face-to-face survey administration, resulting in 600 valid responses.

A structured questionnaire was developed based on established mobile security and privacy studies [20]. It contained demographic questions together with Likert-scale, multiple-response, and binary (Yes/No) items covering permission awareness, permission-management behaviour, privacy experiences, and perceptions of privacy-preserving

technologies. Before the main survey, the questionnaire was pilot-tested with approximately 20 smartphone users, resulting in minor revisions to improve clarity and usability. Both English and Swahili versions were used during data collection to accommodate participants' language preferences [21].

2.4 Data Collection

Multiple data sources were used to support both behavioural analysis and system development. Survey data were collected to assess user awareness, behaviour, and experiences related to Android application permissions. Application metadata were collected to support the development and evaluation of the machine learning model, while controlled experimental data were used to validate the functionality and performance of the PADLOCK prototype.

2.5 Development Framework: Agile (Scrum)

The PADLOCK prototype was developed using an Agile-inspired iterative approach that supported incremental implementation, integration, testing, and refinement throughout the study. Each development iteration included functional validation of newly implemented components before their integration into the complete system, enabling continuous improvement of both system functionality and experimental evaluation. The prototype was developed under supervisory guidance, with iterative testing and validation ensuring the correctness and reliability of the implemented modules before final system evaluation.

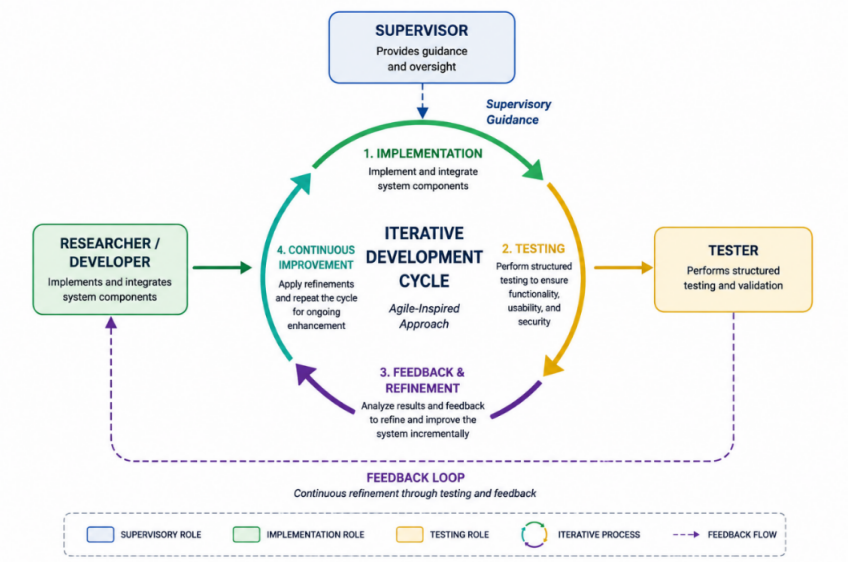

Figure 1: Agile-Inspired PADLOCK System Development and Validation Process

2.6 PADLOCK System Development

This section presents the development of PADLOCK (Privacy Access Dashboard with Logging, Oversight, and Control Kernel), a context-aware Android prototype system designed to monitor application behaviour, assess permission-related privacy risks, and support informed user decision-making. The system integrates Android application-layer monitoring with the embedded ShieldAI machine learning module to provide runtime permission-risk assessment while remaining compatible with the native Android operating system.

PADLOCK collects permission-related events through Android system services, transforms them into feature vectors for the embedded ShieldAI model, and presents the resulting assessments through an interactive user interface. Although the model was developed using financial applications as the case-study domain, the framework is intended for broader Android application categories because it relies on generic permission-based behavioural features.

2.7 PADLOCK System Architecture

This section describes the architecture of the PADLOCK prototype. The system adopts an application-layer architecture that integrates the proposed ShieldAI Compact Deep Neural Network (DNN) with permission monitoring, contextual analysis, user-guided decision support, and audit logging. Hybrid event-driven and scheduled monitoring enables timely permission observation while complying with Android background execution constraints. Consistent with the Principle of Least Privilege (PoLP), PADLOCK focuses on monitoring, analysing, and guiding user responses rather than enforcing system-level controls. Within PADLOCK, the Control Kernel refers to the application-layer coordination logic responsible for monitoring, analysis, and user-mediated responses. It does not modify the Android kernel, ensuring compatibility with standard Android devices. Figure 2 illustrates the overall PADLOCK architecture and operational workflow.

2.8 ShieldAI Module Design Approach

The ShieldAI module was designed to address three key challenges: the absence of independently verified ground-truth labels, the computational constraints of on-device deployment, and the elevated privacy risks associated with Android financial applications.

These considerations informed the selection of heuristic labelling, lightweight neural network architecture, and TensorFlow Lite deployment.

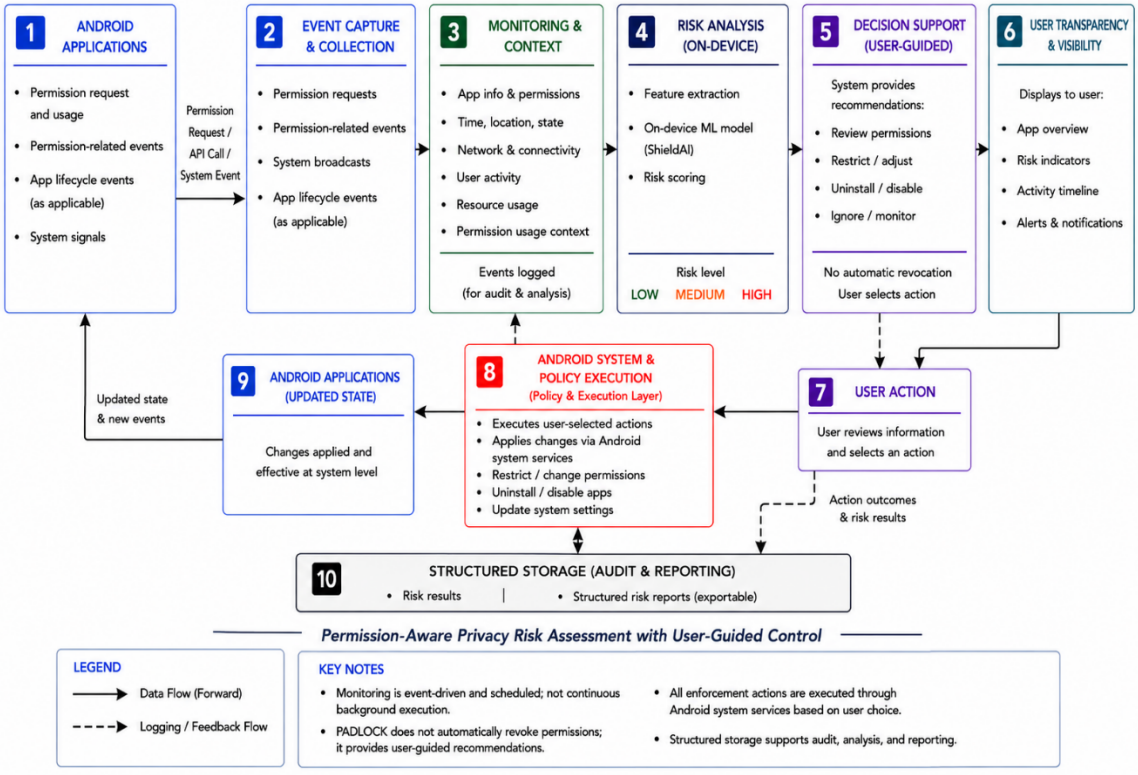

Figure 2: PADLOCK System Operation and Architecture Overview

2.8.1 Problem Formulation

The detection of higher-risk Android applications is formulated as a binary classification problem. Given an input feature vector derived from the permission profile of an Android application, the objective is to learn a classification function as depicted in equation (2).

$$f : \mathbb{R}^d \rightarrow \{0,1\} \tag{2}$$

where:

0 denotes a benign application and

1 denotes a higher-risk application.

Formally, the labelled dataset is represented as shown in equation (3):

$$\mathrm{D} = \{(\mathrm{x_i}, \mathrm{y_i})\}_{\mathrm{i=1}}^{\mathrm{N}} \tag{3}$$

where:

$x_i \in \mathbb{R}^d$ represents a d-dimensional feature vector,

$y_i \in \{0,1\}$ represents the corresponding binary class label and

N denotes the total number of samples.

The classifier is trained by minimising the binary cross-entropy loss function, as presented in equation (4):

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (4)$$

where:

$(\hat{y}_i)$ denotes the predicted probability that sample (i) belongs to the higher-risk class. Because the collected Google Play dataset did not include verified class labels, heuristic risk scoring was first applied to generate binary labels before supervised model training. This heuristic-assisted learning strategy constitutes an integral component of the proposed framework and is described in Section 2.12.

2.8.2 ShieldAI System Architecture

The ShieldAI module is designed as a structured multi-phase pipeline that transforms raw application data into a deployable on-device permission risk assessment model, as illustrated in Figure 3. The architecture comprises four stages:

a) Phase 1: Data preparation and feature engineering
b) Phase 2: Neural network-based risk classification
c) Phase 3: Model serialisation and deployment packaging
d) Phase 4: Integration and deployment within the PADLOCK system

The first three phases constitute the ShieldAI model development pipeline, while the final phase integrates the trained model into PADLOCK for on-device inference and runtime monitoring. Higher-risk labels shown in the architecture are derived from heuristic rule-based annotation and do not represent independently verified malware classifications.

2.9 Design Rationale

Having established the system architecture, this section explains the rationale for the principal design decisions adopted during the development of the ShieldAI module.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

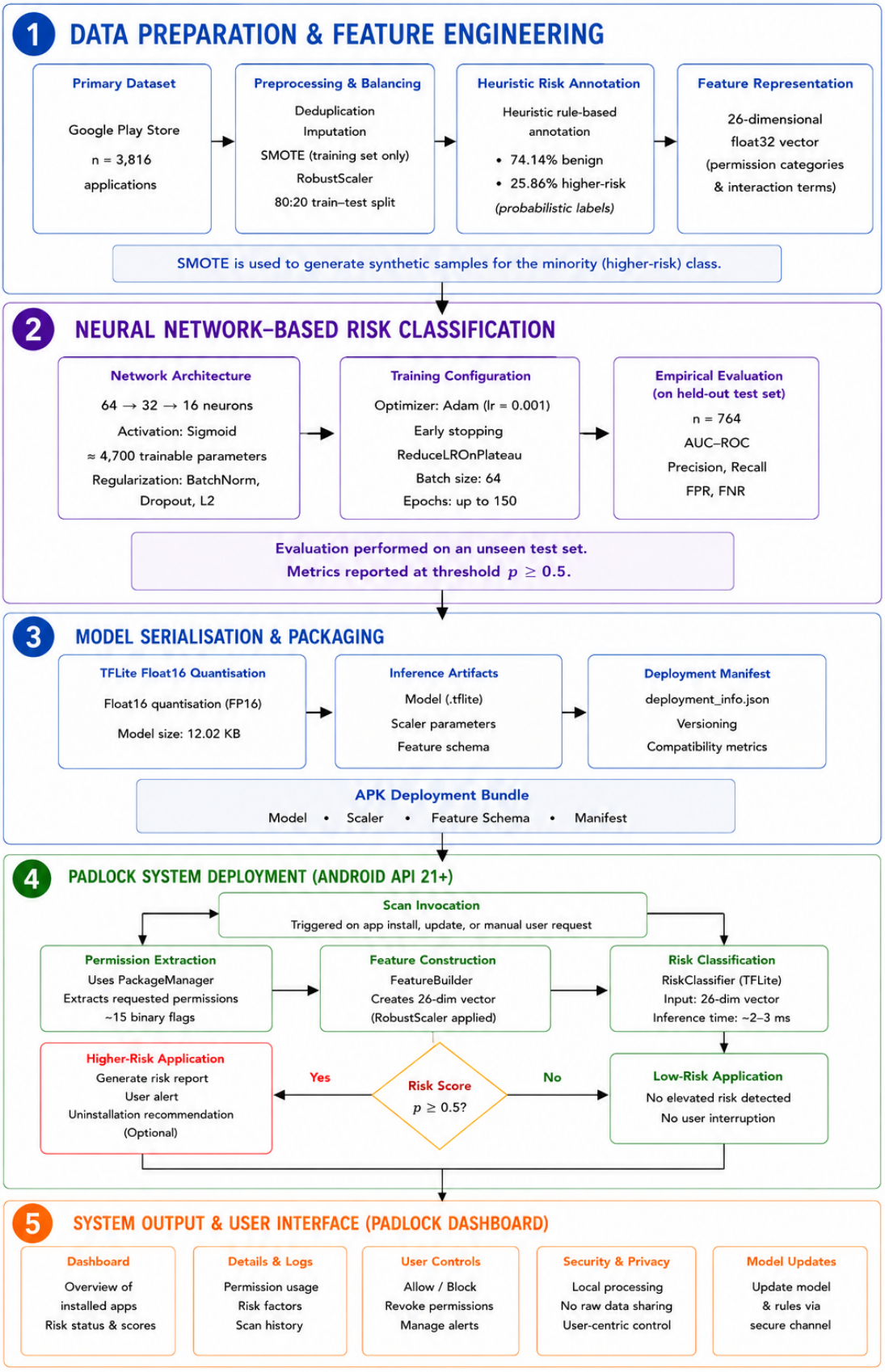

Figure 3. High-Level Architecture of the ShieldAI Prototype Module

2.9.1 Choice of Permission-Based Features

Android permissions provide the primary mechanism through which applications access sensitive device resources and user data. Because permissions are explicitly declared within the application manifest, they provide publicly accessible and semantically meaningful features for analysis. Previous studies have shown that permission-based features are informative for Android risk classification [22]. In financial applications, combinations such as SMS, Contacts, and Internet permissions are associated with elevated privacy and security risks. Furthermore, permission-based analysis supports efficient static assessment without requiring application execution, making the model suitable for lightweight on-device deployment.

2.9.2 Choice of Heuristic Labelling Strategy

The Google Play dataset used in this study did not contain independently verified benign or malware labels. Therefore, an expert-designed heuristic scoring framework was adopted to generate permission-risk labels. This approach is consistent with heuristic-assisted learning methods used in Android security research [23], where domain knowledge is encoded into scoring rules and applications exceeding a predefined threshold are classified as higher-risk. Although heuristic labelling may introduce bias, it provides a practical alternative when verified ground-truth labels are unavailable.

2.9.3 Choice of Neural Network Selection

A compact feedforward neural network was selected to balance predictive performance with efficient mobile deployment. Batch normalisation, dropout, and L2 regularisation were incorporated to improve model generalisation while maintaining a lightweight architecture. The final architecture comprises three hidden layers (64 → 32 → 16 neurons) followed by a sigmoid output layer, resulting in approximately 4,700 trainable parameters suitable for the 26-dimensional feature representation.

2.9.4 Choice of TensorFlow Lite for Deployment

TensorFlow Lite was selected because of its native Android support and efficient on-device inference. Post-training float16 quantisation reduced the deployment size while maintaining comparable predictive performance, enabling low-latency, privacy-preserving execution without cloud dependency.

2.10 Dataset Description

The dataset was compiled from Android application metadata collected from the Google Play Store in January 2025 [24]. The original dataset contained 13,407 applications across multiple categories. The Finance category was selected as the case-study domain because of its elevated privacy and security sensitivity, yielding a working dataset of 3,816 applications [22]. Although model development focused on financial applications, the proposed approach is intended to support broader Android application categories.

Each application record included application metadata together with binary indicators representing 15 Android permission categories, including Camera, Microphone, Location, Contacts, SMS, Phone, Device ID, Storage, Internet, Calendar, and other platform permissions. Because the dataset contained no predefined benign or higher-risk labels, application labels were generated using the heuristic scoring framework described in Section 2.13. The final dataset comprised 2,829 (74.14%) benign and 987 (25.86%) higher-risk applications. In this study, higher-risk refers to applications exhibiting permission patterns associated with elevated privacy or security concerns rather than confirmed malicious behaviour.

2.11 Model Development Workflow

The ShieldAI model was implemented in Python using TensorFlow/Keras within a Jupyter Notebook environment. Development followed a structured workflow comprising data preparation, exploratory analysis, heuristic label generation, feature engineering, model training, evaluation, and TensorFlow Lite conversion for deployment. Standard scientific computing and machine-learning libraries were used for data preprocessing, class balancing, model evaluation, and visualisation [25], while fixed random seeds ensured experimental reproducibility.

The dataset was first filtered to retain financial applications before undergoing preprocessing, including handling missing values, removing duplicate records, validating binary permission indicators, and verifying dataset consistency. Exploratory data analysis was then performed to examine permission distributions and identify patterns associated with higher-risk applications, providing the foundation for the subsequent feature engineering and model development stages [26].

2.12 Label Generation and Feature Engineering

This section describes the heuristic labelling process and feature engineering techniques used to prepare the dataset for model training.

2.12.1 Heuristic-Based Label Generation

Because independently verified malware labels were unavailable, a rule-based heuristic scoring framework was developed to assign a permission-risk score to each application based on its requested permissions. The scoring rules were derived from documented Android permission-usage patterns reported in previous studies [27], [28] and are summarised in Table 1.

Table 1: Heuristic-Based Risk Scoring Rules

Rule	Condition	Score Added
1	SMS ∧ Contacts ∧ Internet	+40
2	SMS ∧ Phone	+35
3	Camera ∧ Contacts ∧ Internet	+30
4	Location ∧ Contacts	+25
5	Microphone requested	+20
6	Calendar requested	+15
7	Number of permissions > 10	+25
7a	Number of permissions > 15	+15 (additive)
8	Number of permissions = 0	+30
9	≥ 3 high-risk permissions present	+20

High-risk permissions were defined as SMS, Contacts, Camera, Microphone, and Location. The scoring framework captures combinations of permissions associated with elevated privacy and security risks. A heuristic threshold of 60 was selected during exploratory analysis to separate benign and higher-risk applications. Applying this threshold to the dataset resulted in 987 (25.86%) higher-risk applications. The higher observed prevalence reflects the greater concentration of sensitive permission combinations within financial and digital lending applications under the adopted heuristic scoring criteria. Each

application was additionally assigned a normalised confidence score (0–1) representing a heuristic estimate of permission-related risk. These labels should be interpreted as heuristic-derived probabilistic indicators rather than independently verified malware ground truth.

2.12.2 Feature Engineering

The original 15-dimensional permission vector was expanded with 11 engineered features, producing a 26-dimensional input representation for the ShieldAI model. These features capture aggregated permission characteristics and higher-order permission interactions that are not explicitly represented by the original permission indicators. The complete feature specification is presented in Table 2.

Table 2: Complete 26-feature input specification for the ShieldAI model

Feature Index	Feature Name	Definition
1–15	Permission flags	Raw binary indicators for each of the 15 permission categories (0 = not requested, 1 = requested)
16	`high_risk_count`	Sum of {SMS, Contacts, Camera, Microphone, Location} flags
17	`moderate_risk_count`	Sum of {Phone, Device ID & call information, Photos/Media/Files} flags
18	`total_permissions`	Sum of all 15 permission flags
19	`risk_ratio`	`high_risk_count/(total_permissions + 1)`
20	`combo_sms_contacts`	SMS × Contacts (interaction term)
21	`combo_sms_internet`	SMS × Internet (interaction term)
22	`combo_contacts_internet`	Contacts × Internet (interaction term)
23	`combo_triple_threat`	SMS × Contacts × Internet (three-way interaction)
24	`combo_camera_internet`	Camera × Internet (interaction term)

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Feature Index	Feature Name	Definition
25	privacy_risk_score	Sum of {SMS, Contacts, Camera, Microphone, Location, Calendar} flags
26	communication_access	Sum of {SMS, Phone, Contacts} flags

2.13 Dataset Preparation

This section describes how the dataset was prepared for training, including splitting and handling class imbalance.

2.13.1 Dataset Partitioning and Class Balancing

The labelled feature-engineered dataset was partitioned into training and test sets using a stratified 80:20 split with a fixed random seed of 42 to preserve class distribution. This produced 3,052 training samples and 764 test samples, comprising 566 benign and 198 higher-risk applications. To address class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) [29] was applied with k=3 to generate synthetic minority-class samples and improve class balance without enforcing complete equalisation [30]. Following class balancing, the training data were normalised using RobustScaler to improve robustness against outliers. To prevent data leakage, the scaler was fitted only on the training data and then applied to the test data using the same transformation during evaluation and deployment.

2.14 Model Design and Training

This section presents the architecture and training process of the ShieldAI model.

2.14.1 Neural Network Architecture

The ShieldAI classifier is implemented as a compact Sequential Feedforward Neural Network using the TensorFlow Keras framework. The architecture consists of the following layers:

a) Input Layer: Accepts the 26-dimensional feature vector described in Table 2.
b) First Hidden Layer: Fully connected with 64 ReLU neurons, followed by batch normalisation, dropout (0.3), and L2 regularisation (0.001) [31].

c) Second Hidden Layer: Fully connected with 32 ReLU neurons, followed by batch normalisation, dropout (0.2), and L2 regularisation (0.001).
d) Third Hidden Layer: Fully connected with 16 ReLU neurons, followed by dropout (0.1) and L2 regularisation (0.001).
e) Output Layer: A single sigmoid neuron produces the probability of an application belonging to the higher-risk class. A decision threshold of 0.5 is used for binary classification.

The model contains approximately 4,700 trainable parameters, making it suitable for lightweight on-device deployment. The complete architecture is summarised in Table 3.

Table 3.: ShieldAI Neural Network Architecture Summary

Layer	Output Shape	Parameters	Regularisation
Input	(None, 26)	0	—
Dense (64, ReLU)	(None, 64)	1,728	L2(0.001)
BatchNormalization	(None, 64)	256	—
Dropout (0.3)	(None, 64)	0	—
Dense (32, ReLU)	(None, 32)	2,080	L2(0.001)
BatchNormalization	(None, 32)	128	—
Dropout (0.2)	(None, 32)	0	—
Dense (16, ReLU)	(None, 16)	528	L2(0.001)
Dropout (0.1)	(None, 16)	0	—
Dense (1, Sigmoid)	(None, 1)	17	—
Total		4,737	

2.14.2Training Configuration

The model was compiled using the Adam optimiser with an initial learning rate of 0.001 and Binary Cross-Entropy loss. During training, accuracy, precision, recall, and AUC were monitored. To address residual class imbalance after SMOTE augmentation, balanced class weights were applied to increase the contribution of minority (higher-risk) samples

during optimisation [32]. Training was performed for a maximum of 150 epochs using a batch size of 32, with 20% of the SMOTE-augmented training data reserved for validation. To improve training efficiency and reduce overfitting, three callback functions were employed:

a) EarlyStopping: Monitored validation loss and restored the best-performing model after 20 epochs without improvement.
b) ReduceLROnPlateau: Reduced the learning rate by half after seven epochs without validation improvement (minimum learning rate: 1×10^{-7}).
c) ModelCheckpoint: Saved the best-performing model based on validation AUC.

Training progress was monitored using accuracy, loss, precision, recall, and AUC, as illustrated in Figure 8. Although model development used financial applications as the case-study domain, the extracted permission-based features are not domain-specific, supporting potential applicability across broader Android application categories.

2.14.3Model Validation and Evaluation Metrics

After training, the model was evaluated using a held-out test set comprising 764 applications (566 benign and 198 higher-risk), which were excluded from model training. Probabilistic predictions were converted to binary classifications using a decision threshold of 0.5, and performance was assessed using standard confusion-matrix-based evaluation metrics [33]. The evaluation metrics summarised in Table 4 provide a comprehensive assessment of the model's predictive performance, error distribution, and ability to distinguish benign from higher-risk Android applications.

Table 4: Confusion Matrix Components Used for Model Evaluation

Symbol	Description
TP (True Positives)	Higher-risk applications are correctly classified as higher-risk
TN (True Negatives)	Benign applications are correctly classified as benign
FP (False Positives)	Benign applications are incorrectly classified as higher-risk
FN (False Negatives)	Higher-risk applications incorrectly classified as benign

a) Accuracy

Accuracy measures the overall proportion of correctly classified applications, indicating the general effectiveness of the model, as defined in equation (5).

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{5}$$

b) Precision

Precision measures the proportion of predicted higher-risk applications that are correctly classified (Equation 6).

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{6}$$

c) Recall (Sensitivity)

Recall measures the proportion of actual higher-risk applications correctly identified by the model (Equation 7).

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{7}$$

d) F1-Score

The F1-score combines precision and recall into a single performance measure, particularly useful for imbalanced datasets (Equation 8).

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \tag{8}$$

e) Receiver Operating Characteristic (ROC) Curve

The ROC curve evaluates the trade-off between the true positive rate (TPR) and false positive rate (FPR), computed using Equations (10) and (11). The true positive rate (TPR) is computed as defined in Equation (9).

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}} \tag{9}$$

The false positive rate (FPR) is computed as presented in Equation (10).

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} \tag{10}$$

f) Area Under the Curve (ROC-AUC)

ROC-AUC summarises the model's ability to distinguish between benign and higher-risk applications across all decision thresholds.

g) Precision–Recall Curve and PR-AUC (Average Precision)

The Precision–Recall curve and its corresponding area under the curve (PR-AUC) provide additional evaluation of positive-class performance under class imbalance. In addition to these metrics, the confusion matrix and prediction probability distributions (Figure 9) were analysed to assess class separability. Collectively, these complementary measures provide a comprehensive evaluation of model performance under imbalanced classification conditions.

2.15 TFLite Conversion and Deployment Packaging

Following model training, the ShieldAI model was prepared for deployment through model serialisation and TensorFlow Lite conversion.

2.15.1 Model Serialization and TFLite Conversion

Following model training, the complete Keras model was retained to support future retraining and refinement. For deployment, the trained model was converted to TensorFlow Lite using post-training float16 quantisation, reducing the model size while maintaining comparable predictive performance. The converted model occupied approximately 12 KB, making it suitable for deployment on resource-constrained Android devices.

2.15.2 TFLite Inference Verification

The converted TensorFlow Lite model was evaluated against the original Keras model using the held-out test set to verify inference consistency. Inference latency was further assessed over 100 predictions using the mean and 95th-percentile response times. The final deployment package comprises the artefacts summarised in Table 5, together with deployment metadata describing the model version, training information, dataset characteristics, and performance metrics required for on-device deployment.

Table 5: ShieldAI Deployment Artefacts

Artefact	Description
`shieldai_model.tflite`	Quantised TensorFlow Lite deployment model (~12 KB)
`shieldai_scaler.pkl`	Feature normalisation model
`deployment_info.json`	Deployment metadata and model information

Following verification and packaging, the ShieldAI model was integrated into the PADLOCK system to support on-device permission-risk assessment.

2.16 PADLOCK System Integration and Data Flow

This section describes how the trained ShieldAI model is integrated into PADLOCK and how permission-related information is processed during runtime. PADLOCK integrates the embedded TensorFlow Lite ShieldAI model with Android application monitoring through Flutter platform channels (MethodChannel and EventChannel), connecting the Flutter application layer and native Kotlin services [34]. During runtime, permission usage, application events, and network activity are collected and transformed into the feature representation used during model development. The feature vector is subsequently normalized and processed by the embedded ShieldAI model to generate a probabilistic risk score for classifying applications as benign or higher-risk [35]. The generated assessments are presented through risk indicators, contextual alerts, and user-guided recommendations, while audit records are maintained locally for reporting. All processing is performed on the device to preserve user privacy.

2.17 System Technology Stack

The PADLOCK prototype was implemented using a combination of Android and machine learning technologies. The principal implementation technologies are summarised below.

Table 6: PADLOCK Implementation Technology Stack

Technology	Purpose
Flutter (Dart)	User interface and application logic
Kotlin	Native Android monitoring services

Technology	Purpose
TensorFlow Lite	On-device ShieldAI inference
Android WorkManager	Scheduled background tasks
Android AppOps API	Permission usage monitoring
NetworkStatsManager	Per-application network statistics
AES-256 and SHA-256	Secure storage and integrity protection

2.18 PADLOCK System Testing and Validation

The PADLOCK prototype was evaluated under controlled Android execution conditions to verify the integration of the embedded ShieldAI model and its ability to monitor application behaviour, process permission-related information, and generate runtime permission-risk assessments.

2.18.1 Experimental Setup

The PADLOCK system was evaluated under controlled conditions using both Android emulators and physical devices. The runtime deployment environment consisted of a Samsung Galaxy S23 and the Android Studio Emulator, both running Android 12 or later. The mobile application was developed using Flutter (Dart) with native Kotlin components and executed on a device equipped with 8 GB RAM and an octa-core Snapdragon processor. The machine learning model was developed and trained in Jupyter Notebook using TensorFlow. Deployment was carried out using TensorFlow Lite to enable on-device inference. Model training was performed on a workstation equipped with 8–16 GB RAM and 4–8 physical CPU cores, utilizing a dataset of 3,816 Android applications. Performance evaluation was conducted on a held-out test set comprising 764 applications. This experimental configuration enabled evaluation of system functionality, permission monitoring, behavioural observation, and ShieldAI model integration under controlled execution conditions.

2.19 Operational and Performance Considerations

PADLOCK operates within standard Android security constraints, where access to permission usage requires explicit user authorisation and background execution is restricted by the operating system. To accommodate these limitations, the system combines event-driven monitoring with lightweight scheduled tasks, while detailed

network inspection is limited to high-level per-application usage statistics because deeper traffic analysis requires elevated privileges. The embedded TensorFlow Lite model enables efficient on-device inference with minimal computational overhead, supporting deployment on resource-constrained Android devices. Although detailed energy measurements were outside the scope of this study, the system was designed to minimise resource consumption while maintaining timely permission monitoring and runtime risk assessment.

2.20 Ethical Considerations and Bias Mitigation

Ethical principles were observed throughout the study. Participation in the survey was voluntary, informed consent was obtained from all participants, and no personally identifiable information was collected. The machine learning component used only publicly available Google Play Store metadata without accessing personal user data, private communications, or application binaries. The study complied with institutional ethical requirements and the Tanzania Personal Data Protection Act, 2022 [36]. To reduce potential bias, a consistent rule-based labelling framework was adopted, and model evaluation was performed using an independent test set. These measures support the reliability of the reported findings. Despite these measures, the use of heuristic labelling may introduce inherent bias. This limitation is acknowledged and considered in the interpretation of the results.

3 RESULTS AND DISCUSSION

This chapter presents the experimental findings from the evaluation of the proposed PADLOCK prototype system. The results include survey findings, model performance, system implementation outcomes, and behavioural evaluation, followed by a discussion of their implications for Android permission-risk assessment.

3.1. Respondent Population Characteristics

This section presents the survey findings on permission-related experiences and the usability evaluation of the proposed PADLOCK system.

1) Survey Results

The questionnaire included multiple items assessing Android permission use, privacy awareness, and security experiences. Table 7 summarises responses to four representative questionnaire items (Q25, Q28, Q31, and Q34).

Table 7: Reported Permission-Related Experiences Among Smartphone Users (n = 600)

Survey Item	Responses (n)	Percentage (%)
Experienced privacy or security issues associated with application permission requests (Q25)	295	64.7
Experienced or were aware of loan applications accessing sensitive data without explicit consent (Q28)	293	64.3
Experienced or were aware of humiliation or reputational harm associated with loan-application practices (Q31)	318	69.7
Noticed gaming applications accessing sensitive information or collecting data for non-game purposes (Q34)	246	53.9

As shown in Table 7, the highest proportion of respondents (69.7%) reported experiencing or being aware of humiliation or reputational harm associated with loan-application practices. Privacy or security issues related to application permission requests and unauthorised access to sensitive information by loan applications were reported by 64.7% and 64.3% of respondents, respectively. In addition, 53.9% of respondents indicated that gaming applications accessed sensitive information or collected data for purposes unrelated to gameplay.

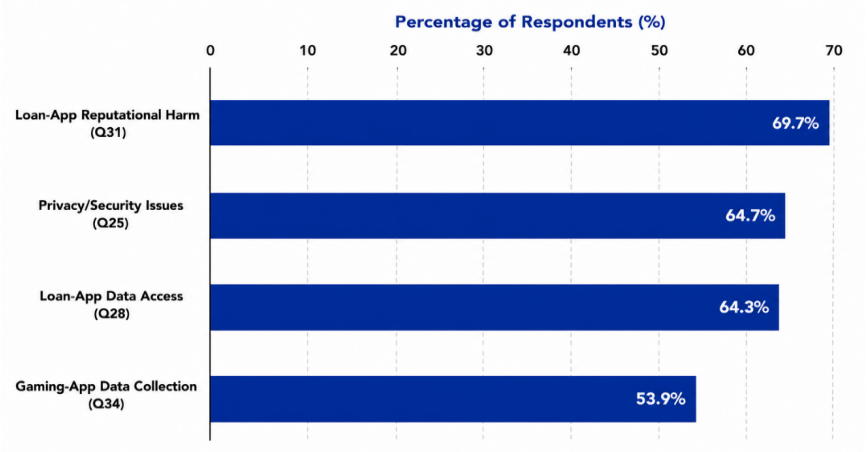

Figure 4: Percentage distribution of reported permission-related experiences among smartphone users.

2) Usability Evaluation (User Acceptance Testing)

Thirty participants evaluated PADLOCK using the System Usability Scale (SUS). The system achieved an average SUS score of 78/100, exceeding the accepted benchmark of 68, indicating good overall usability for permission monitoring and risk communication.

SUS Score	Perticipants
0-69	5
70-79	12
80-89	9
90-100	4

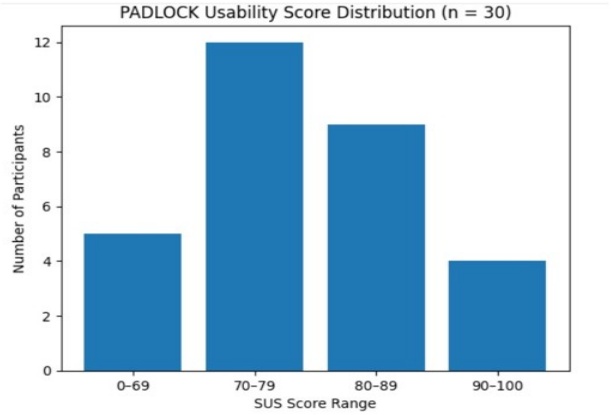

Figure 5: Distribution of SUS Usability Score

3.2. Exploratory Model Data Analysis

This section presents exploratory analyses of the cleaned dataset, including permission distributions, permission co-occurrence patterns, and application metadata used to support feature engineering and model development.

1) Permission Distribution Analysis

Figure 6 summarises the distribution of fifteen permission categories across 3,816 financial applications. Permission counts exhibited a right-skewed distribution with a mean of 6.18, a median of 6, and an interquartile range of 6 (Q1 = 3, Q3 = 9). The Internet permission appeared in nearly all applications, followed by Photos/Media/Files, Storage, and Camera permissions. A smaller proportion of applications request permissions associated with SMS, Microphone, and Contacts access. Applications requesting more than 14 permissions were identified as statistical outliers using the IQR criterion. These observations informed the feature engineering process adopted by the ShieldAI model.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

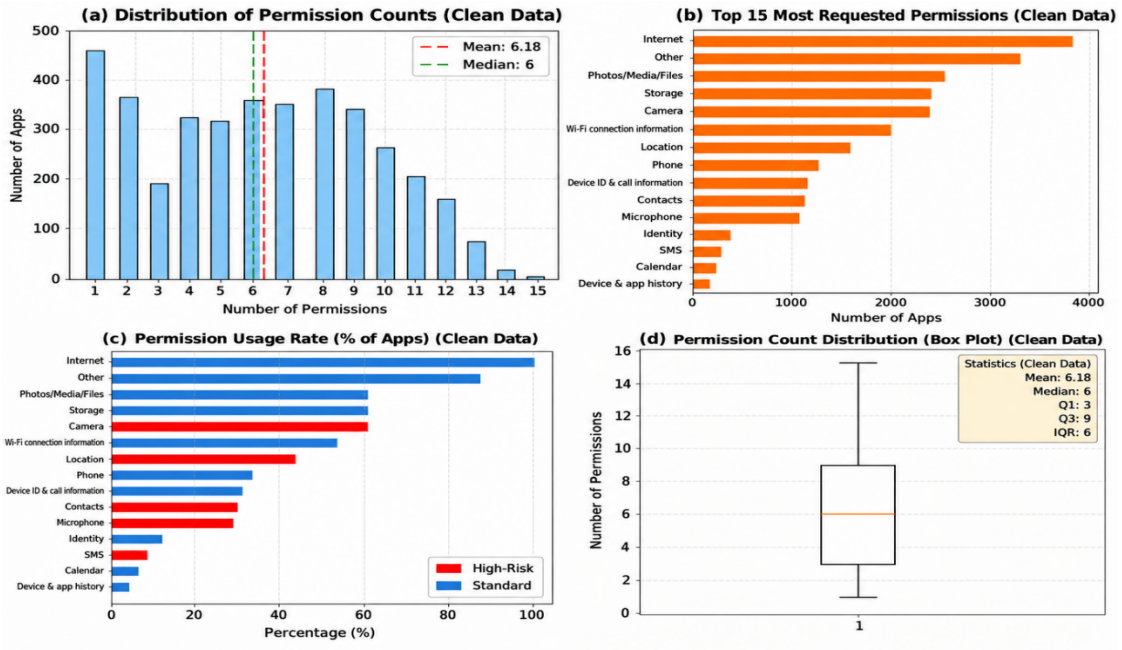

Figure 6: Permission distribution and data analysis for 3,816 financial applications, including: (a) histogram of permission counts per application with mean (6.18) and median (6); (b) horizontal bar chart of the top 15 permission categories; (c) permission usage rates with high-risk permissions highlighted; and (d) box plot of permission count distribution (Q1 = 3, median = 6, Q3 = 9, IQR = 6).

2) Permission Co-Occurrence Analysis

A Pearson correlation matrix was computed for nine permission categories: SMS, Contacts, Camera, Microphone, Location, Phone, Internet, Storage, and Photos/Media/Files. Figure 7 shows several moderate positive permission co-occurrence relationships, with the strongest observed between Camera and Location ($r = 0.53$), Camera and Photos/Media/Files ($r = 0.50$), Camera and Storage ($r = 0.50$), Camera and Microphone ($r = 0.48$), and Location and Phone ($r = 0.46$). The Internet permission produced undefined (NaN) correlation coefficients with all other permissions, indicating uniform occurrence across all applications within the dataset. The observed co-occurrence patterns informed the feature engineering process used during model development.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Figure 7: Pearson correlation matrix for permission categories

3.3. Model Training and Validation

Following the methodology described in Section 2.15, the ShieldAI model was trained and evaluated under controlled experimental conditions. Figure 8 illustrates the training behaviour of the model across training epochs using accuracy, loss, AUC, precision, and recall metrics. Training and validation accuracy exceeded 0.95 within the initial epochs. The binary cross-entropy loss decreased progressively for both training and validation datasets throughout the training process. The training and validation curves remained closely aligned across epochs. In addition, the AUC stabilised at approximately 0.98 during training. Precision and recall similarly converged to high values across successive epochs. The training process was completed after 62 epochs using early stopping, as illustrated in Figure 8. Following successful training, the model's performance is further evaluated using unseen test data, as described in the following section.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

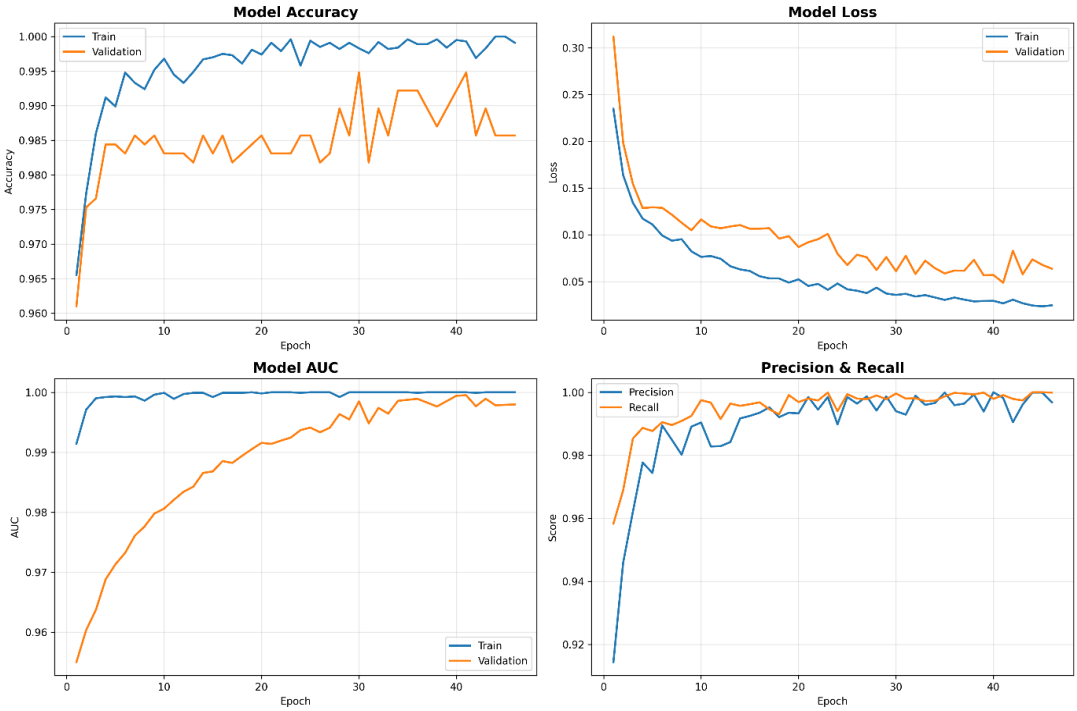

Figure 8: ShieldAI Neural Network Training Across 62 Epochs (Early Stopping)

3.4. Model Evaluation

Using the evaluation framework described in Subsection 2.15.3 and referring to Table 8 and Figure 9, the performance of the ShieldAI model was evaluated using a held-out test dataset (n = 764). The model achieved an accuracy of 96.73%, with precision and recall values of 94.82% and 92.42%, respectively, resulting in an F1-score of 93.61%. The ROC-AUC and PR-AUC values were 0.9824 and 0.9631, respectively [37].

The confusion matrix presented in Figure 9(a) shows 556 true negatives and 183 true positives, with 10 false positives and 15 false negatives recorded during evaluation. The ROC curve illustrated in Figure 9(b) produced an AUC value of 0.9824. The precision–recall curve shown in Figure 9(c) produced an average precision value of 0.9631. The probability distribution presented in Figure 9(d) illustrates the distribution of benign and higher-risk application classes within the evaluated dataset [38].

Table 8: ShieldAI Model Performance Evaluation Results

Accuracy	Precision	Recall	F1-Score	PR-AUC	ROC-AUC	TN	FP	FN	TP
0.9673 (96.73%)	0.9482 (94.82%)	0.9242 (92.42%)	0.9361 (93.61%)	0.9631	0.9824	556	10	15	183

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

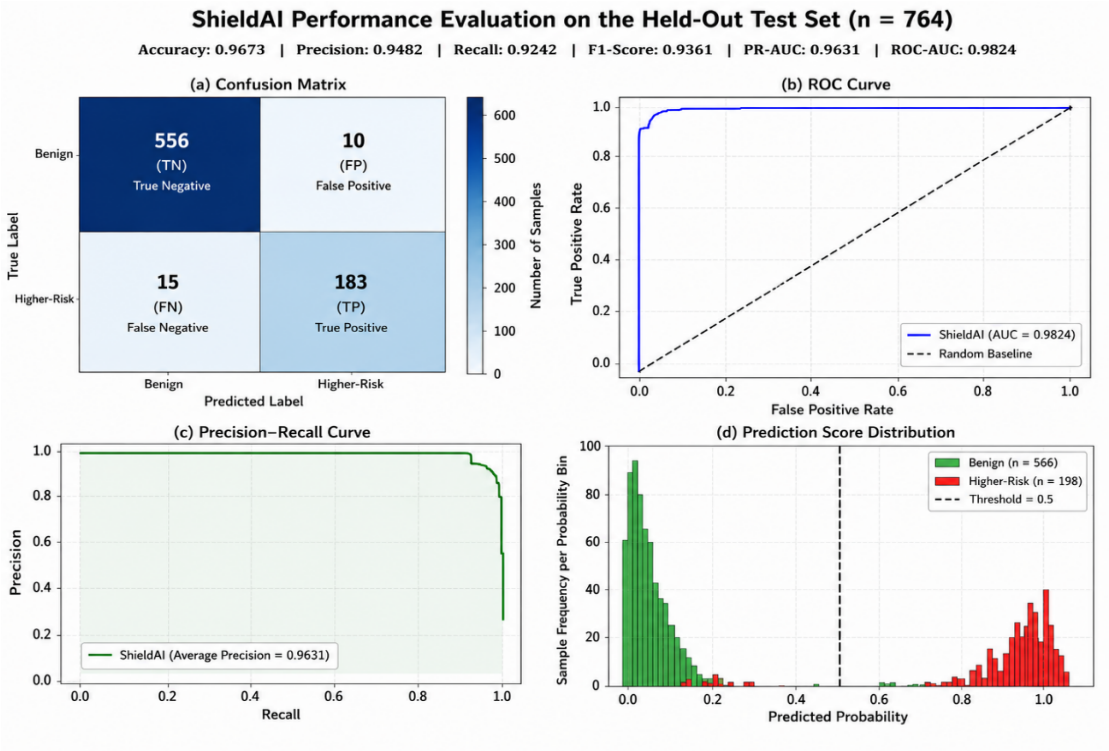

Figure 9: Model evaluation results on the held-out test set (n = 764)

3.5. Comparative Analysis of the ShieldAI Compact DNN Against Baseline Models

The proposed ShieldAI Compact DNN was compared with a Standard Deep DNN and a Random Forest classifier under identical experimental conditions [39]. The Standard Deep DNN achieved the highest classification performance (F1-score = 0.9487; ROC-AUC = 0.9891), whereas ShieldAI achieved comparable performance (F1-score = 0.9361; ROC-AUC = 0.9824) while maintaining a substantially smaller deployment size (~12 KB) and lower inference latency (approximately 2–3 ms) [40]. The compact size of ShieldAI was achieved through an intentionally lightweight network architecture with fewer trainable parameters, followed by post-training float16 TensorFlow Lite quantisation, enabling efficient on-device deployment without substantial loss of predictive performance. In contrast, the Random Forest classifier achieved the lowest classification performance (F1-score = 0.8950) and required a considerably larger model (372.4 KB) with slower inference. These findings demonstrate that ShieldAI provides the best balance between predictive performance and deployment efficiency for resource-constrained Android devices.

3.5.1. Implementation Outcomes of the PADLOCK Interface System

This section presents the implementation outcomes following integration of the ShieldAI model within the PADLOCK Android environment.

1) System Deployment and Runtime Performance

The PADLOCK interface provides event-driven permission monitoring, contextual risk reporting, alert generation, permission review, and audit-oriented reporting functions within the Android environment [41], [42]. During on-device evaluation, PADLOCK maintained the classification performance observed during model evaluation on the held-out test dataset while sustaining an average inference latency below 2.6 ms per prediction.

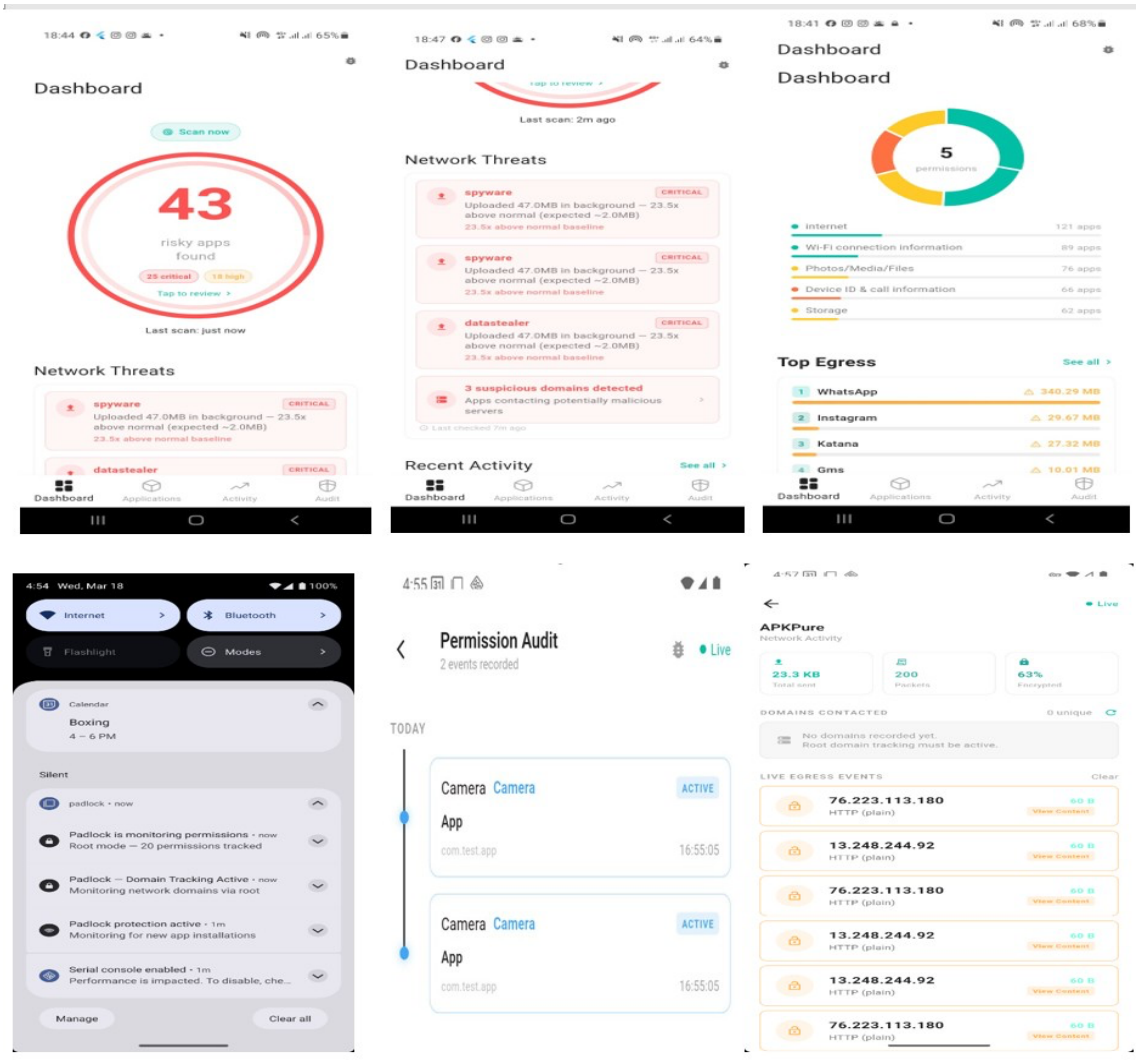

Figure 10: Collective illustration of PADLOCK's security dashboard

2) Behavioural Impact Assessment

A controlled behavioural evaluation involving 30 participants was conducted to compare responses to higher-risk permission request scenarios before and after interaction with PADLOCK. During the baseline phase, participants interacted with Android applications containing predefined higher-risk permission request scenarios without PADLOCK assistance. Accepted higher-risk permission grants were recorded according to the heuristic risk criteria defined in Sections 2.12. During the intervention phase, the same participants repeated comparable tasks with PADLOCK enabled, which provided contextual permission alerts, visibility indicators, and explanatory guidance before permission decisions were made. The percentage reduction in higher-risk permission grants was calculated by comparing the baseline and PADLOCK-assisted conditions. Compared with the baseline condition, PADLOCK-assisted interaction reduced higher-risk permission grants by approximately 88% under the evaluated experimental conditions.

3.6. Discussion

This section discusses the findings in relation to Android permission-risk assessment, lightweight on-device deployment, and the practical implications of the proposed PADLOCK approach.

3.6.1. Interpretation of Model Performance

The evaluation results indicate that permission-based behavioural features provide effective discriminative information for identifying higher-risk Android applications within the adopted heuristic framework. The strong performance achieved across Accuracy, Precision, Recall, F1-score, ROC-AUC, and PR-AUC demonstrates that the selected permission-related features effectively captured meaningful permission-risk patterns. Furthermore, the stable training behaviour observed during model development indicates stable convergence with limited overfitting under the evaluated conditions.

Comparison with the baseline models further highlights the trade-off between predictive performance and deployment efficiency. Although the Standard Deep DNN achieved slightly higher classification performance, the ShieldAI Compact DNN offered a more practical balance between predictive capability, lightweight architecture, and low inference latency, making it well-suited for resource-constrained Android devices.

Unlike many existing Android malware detection approaches that primarily emphasise offline classification [43], [44], [45], [46], [47], [48], PADLOCK combines lightweight machine learning with contextual permission visibility, event-driven monitoring, and user-guided decision support within an on-device Android environment. Nevertheless, direct comparison with previous studies should be interpreted cautiously because of differences in datasets, feature engineering approaches, heuristic labelling strategies, and evaluation settings. Consequently, the reported results reflect permission-risk assessment performance within the adopted evaluation framework rather than definitive malware detection capability.

3.6.2. Behavioural and Usability Implications

The survey findings indicate that permission-related privacy concerns remain widespread among smartphone users, particularly in applications supporting communication, financial transactions, and digital services. These observations are consistent with previous studies reporting limited user awareness of Android permission practices and their associated privacy risks [49]. The behavioural evaluation indicates that contextual permission explanations and risk guidance may positively influence user decision-making. The observed reduction in higher-risk permission grants during PADLOCK-assisted interaction indicates that contextual risk communication may encourage more informed permission decisions, while the usability results demonstrate that these capabilities can be delivered without compromising user experience [50]. These findings should, however, be interpreted within the limitations of the controlled experimental setting, limited participant sample, and heuristic-based risk definitions. Consequently, they reflect behavioural tendencies observed under the evaluated conditions rather than broadly generalisable user behaviour.

3.6.3. Practical Implications of the PADLOCK System

The findings demonstrate that lightweight permission-risk assessment can be integrated into Android applications while maintaining low runtime overhead. By combining on-device inference, contextual permission monitoring, and user-guided decision support, PADLOCK demonstrates a practical approach for improving permission transparency and privacy awareness within Android environments. However, the findings should be interpreted within the heuristic evaluation framework and controlled experimental setting adopted in this study.

4. CONCLUSION

This study proposed PADLOCK, a context-aware Android permission-risk assessment prototype that integrates lightweight machine learning, event-driven monitoring, contextual permission visibility, and user-guided decision support. The evaluation demonstrated that the ShieldAI model achieved strong permission-risk assessment performance while maintaining low computational overhead suitable for on-device deployment. The behavioural evaluation further suggests that contextual permission guidance can support more informed user decision-making during application interaction. Although the findings should be interpreted within the heuristic evaluation framework adopted in this study, the results demonstrate the practical potential of lightweight, context-aware permission-risk assessment for improving permission transparency, contextual risk visibility, and privacy-oriented decision-making within Android environments.

ACKNOWLEDGEMENTS

I sincerely thank Dr. Bonny Mgawe and Dr. Judith Leo for their valuable guidance and support throughout this study. I also extend my appreciation to Nelson Mandela African Institution of Science and Technology (NM-AIST) for providing the necessary environment and resources. I am also grateful to my family, friends, and all participants whose contributions made this study possible.

REFERENCES

[1] C. Liu, J. Lu, W. Feng, E. Du, L. Di, and Z. Song, "MOBIPCR: Efficient, accurate, and strict ML-based mobile malware detection," *Future Generation Computer Systems*, vol. 144, pp. 140–150, Jul. 2023, doi: 10.1016/j.future.2023.02.014.

[2] S. Yilmaz and M. Davis, "Hidden Permissions on Android: A Permission-Based Android Mobile Privacy Risk Model," *European Conference on Cyber Warfare and Security*, vol. 22, no. 1, pp. 717–724, Jun. 2023, doi: 10.34190/eccws.22.1.1453.

[3] G. S. Tuncay, "Android Permissions: Evolution, Attacks, and Best Practices," *IEEE Secur. Priv.*, vol. 22, no. 6, pp. 40–49, 2024, doi: 10.1109/MSEC.2024.3461629.

[4] D. Hayes, F. Cappa, and N. A. Le-Khac, "An effective approach to mobile device management: Security and privacy issues associated with mobile applications," *Digital Business*, vol. 1, no. 1, p. 100001, Sep. 2020, doi: 10.1016/j.digbus.2020.100001.

[5] B. Mishra *et al.*, "Privacy Protection Framework for Android," *IEEE Access*, vol. 10, pp. 7973–7988, 2022, doi: 10.1109/ACCESS.2022.3142345.

[6] O. A. Akanji, M. Egele, and G. Stringhini, "The Cost of Convenience: Identifying, Analyzing, and Mitigating Predatory Loan Applications on Android," *in Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS '26), Bangalore, India*, Jun. 2026, doi: 10.1145/3779208.3785263.

[7] C. W. Munyendo, Y. Acar, and A. J. Aviv, "'Desperate Times Call for Desperate Measures': User Concerns with Mobile Loan Apps in Kenya," *Proc. IEEE Symp. Secur. Priv.*, pp. 2304–2319, May 2022, doi: 10.1109/SP46214.2022.9833779.

[8] E. Caushaj and V. Sugumaran, "Classification and security assessment of Android apps," *Discover Internet of Things*, vol. 3, p. 15, Oct. 2023, doi: 10.1007/S43926-023-00047-0.

[9] Tanzania Communications Regulatory Authority (TCRA), "Communications Statistics Report," Sep. 2025, Accessed: Mar. 06, 2026. [Online]. Available: https://www.tcra.go.tz/services/statistics

[10] M. Khedkar, A. Kumar Mondal, and E. Bodden, "A study of privacy-related data collected by Android apps," *Automated Software Engineering*, vol. 33, no. 2, p. 45, Jan. 2026, doi: 10.1007/s10515-025-00589-3.

[11] Bank of Tanzania and Tanzania Communications Regulatory Authority, "Regulatory Notice on Unlicensed Digital Lending Applications," Nov. 2024, Accessed: Mar. 06, 2026. [Online]. Available: https://www.bot.go.tz/PressRelease

[12] I. M. Almomani and A. Al Khayer, "A Comprehensive Analysis of the Android Permissions System," *IEEE Access*, vol. 8, pp. 216671–216688, Nov. 2020, doi: 10.1109/ACCESS.2020.3041432.

[13] W. Wang, C. Ren, H. Song, S. Zhang, and P. Liu, "FGL_Droid: An Efficient Android Malware Detection Method Based on Hybrid Analysis," *Security and Communication Networks*, vol. 2022, no. 1, p. 8398591, Jan. 2022, doi: 10.1155/2022/8398591.

[14] V. Ayres-Pereira, A. Pirrone, M. Korbmacher, I. Tjostheim, and G. Böhm, "The privacy and control paradoxes in the context of smartphone apps," *Front. Comput. Sci.*, vol. 4, p. 986138, Sep. 2022, doi: 10.3389/fcomp.2022.986138.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[15] S. S. Bakare, A. O. Adeniyi, C. U. Akpuokwe, and N. E. Eneh, "Data Privacy Laws And Compliance: A Comparative Review Of The Eu Gdpr And Usa Regulations," *Computer Science & IT Research Journal*, vol. 5, no. 3, pp. 528–543, Mar. 2024, doi: 10.51594/csitrj.v5i3.859.

[16] A. Lekshmi, J. Atul, A. J. Pillai, M. Dhanya, and S. Kaladharan, "Risks in Instant Loan Apps: Analyzing User Perceptions Using Machine Learning Approach," in *ICT Analysis and Applications, S. Fong, N. Dey, and A. Joshi, Eds., Lecture Notes in Networks and Systems, vol. 1161. Singapore*, Springer, Mar. 2025, pp. 461–471. doi: 10.1007/978-981-97-8602-2_41.

[17] O. Haggag, A. Pedace, S. Pan, and J. Grundy, "An analysis of privacy regulations and user concerns of finance mobile applications," *Inf. Softw. Technol.*, vol. 184, Aug. 2025, doi: 10.1016/j.infsof.2025.107756.

[18] S. D. Minc, P. P. Chandanabhumma, C. L. Sedney, T. S. Haggerty, D. M. Davidov, and R. A. Pollini, "Mixed methods research: A primer for the vascular surgeon," *Semin. Vasc. Surg.*, vol. 35, no. 4, pp. 447–455, Dec. 2022, doi: 10.1053/j.semvascsurg.2022.09.003.

[19] T. Yamane, *Statistics: An Introductory Analysis*, 2nd ed. New York: Harper and Row, 1967.

[20] F. Breitinger, R. Tully-Doyle, and C. Hassenfeldt, "A survey on smartphone user's security choices, awareness and education," *Comput. Secur.*, vol. 88, p. 101647, Jan. 2020, doi: 10.1016/J.COSE.2019.101647.

[21] M. Sandesara *et al.*, "Design and Experience of Mobile Applications: A Pilot Survey," *Mathematics*, vol. 10, no. 14, p. 2380, Jul. 2022, doi: 10.3390/MATH10142380.

[22] S. Maganur, Y. Jiang, J. Huang, and F. Zhong, "Feature-Centric Approaches to Android Malware Analysis: A Survey," *Computers*, vol. 14, no. 11, p. 482, Nov. 2025, doi: 10.3390/computers14110482.

[23] P. K. Mvula, P. Branco, G. V. Jourdan, and H. L. Viktor, "A Survey on the Applications of Semi-supervised Learning to Cyber-security," *ACM Comput. Surv.*, vol. 56, no. 10, Jun. 2024, doi: 10.1145/3657647.

[24] Google, "Android Apps on Google Play," Google Play Store. Accessed: Jan. 20, 2025. [Online]. Available: https://play.google.com/store/apps

[25] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng, "Review of Classification Methods on Unbalanced Data Sets," *IEEE Access*, vol. 9, pp. 64606–64628, Apr. 2021, doi: 10.1109/ACCESS.2021.3074243.

[26] A. Kaur, S. Lal, S. Goel, M. Pandey, and A. Agarwal, "Android Malware Detection System using Machine Learning," *ACM International Conference Proceeding Series*, vol. 1, pp. 186–191, Oct. 2024, doi: 10.1145/3675888.3676049.

[27] S. Lee and S. Han, "SMAD: Semi-Supervised Android Malware Detection via Consistency on Fine-Grained Spatial Representations," *Electronics (Basel)*, vol. 14, no. 21, p. 4246, Oct. 2025, doi: 10.3390/electronics14214246.

[28] A. Muzaffar, H. R. Hassen, H. Zantout, and M. A. Lones, "ActDroid: An active learning framework for android malware detection," *Comput. Secur.*, vol. 160, Jan. 2026, doi: 10.1016/j.cose.2025.104724.

[29] D. Elreedy *et al.*, "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning," *Mach. Learn.*, vol. 113, no. 7, pp. 4903–4923, Jan. 2023, doi: 10.1007/S10994-022-06296-4.

[30] J. Sadaiyandi, P. Arumugam, A. K. Sangaiah, and C. Zhang, "Stratified Sampling-Based Deep Learning Approach to Increase Prediction Accuracy of Unbalanced Dataset," *Electronics (Basel)*, vol. 12, no. 21, p. 4423, Oct. 2023, doi: 10.3390/electronics12214423.

[31] I. Salehin and D. K. Kang, "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain," *Electronics (Basel)*, vol. 12, no. 14, p. 3106, Jul. 2023, doi: 10.3390/electronics12143106.

[32] S. Song and S. Yang, "TS-SMOTE: An Improved SMOTE Method Based on Symmetric Triangle Scoring Mechanism for Solving Class-Imbalanced Problems," *Symmetry (Basel)*, vol. 17, no. 8, p. 1326, Aug. 2025, doi: 10.3390/SYM17081326.

[33] J. C. Obi, "A comparative study of several classification metrics and their performances on data," *World Journal of Advanced Engineering Technology and Sciences*, vol. 8, no. 1, pp. 308–314, Feb. 2023, doi: 10.30574/wjaets.2023.8.1.0054.

[34] Flutter Developers, "Platform Channels: Writing Custom Platform-Specific Code," Flutter Documentation. Accessed: Apr. 25, 2026. [Online]. Available: https://docs.flutter.dev/platform-integration/platform-channels

[35] A. Mahindru *et al.*, "PermDroid a framework developed using proposed feature selection approach and machine learning techniques for Android malware detection," *Sci. Rep.*, vol. 14, no. 1, p. 10724, May 2024, doi: 10.1038/s41598-024-60982-y.

[36] A. Nandan Prasad, "Ethical Implications and Bias Mitigation," in *Introduction to Data Governance for Machine Learning Systems: Fundamental Principles, Critical*

Practices, and Future Trends, A. Nandan Prasad, Ed., Berkeley, CA: Apress, 2024, pp. 307–382. doi: 10.1007/979-8-8688-1023-7_5.

[37] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics (Basel)*, vol. 10, no. 13, p. 1606, Jul. 2021, doi: 10.3390/electronics10131606.

[38] A. R. Nasser, A. M. Hasan, and A. J. Humaidi, "DL-AMDet: Deep learning-based malware detector for android," *Intelligent Systems with Applications*, vol. 21, p. 200318, Mar. 2024, doi: 10.1016/J.ISWA.2023.200318.

[39] O. N. Elayan and A. M. Mustafa, "Android Malware Detection Using Deep Learning," *Procedia Comput. Sci.*, vol. 184, pp. 847–852, Jan. 2021, doi: 10.1016/J.PROCS.2021.03.106.

[40] M. Ibrahim, B. Issa, and M. B. Jasser, "A Method for Automatic Android Malware Detection Based on Static Analysis and Deep Learning," *IEEE Access*, vol. 10, pp. 117334–117352, 2022, doi: 10.1109/ACCESS.2022.3219047.

[41] N. Topalli and A. Badii, "A User-Centric Context-Aware Framework for Real-Time Optimisation of Multimedia Data Privacy Protection, and Information Retention Within Multimodal AI Systems," *Sensors*, vol. 25, no. 19, p. 6105, Oct. 2025, doi: 10.3390/s25196105.

[42] J. L. Herrera, H. Y. Chen, J. Berrocal, J. M. Murillo, and C. Julien, "Context-aware privacy-preserving access control for mobile computing," *Pervasive Mob. Comput.*, vol. 87, p. 101725, Dec. 2022, doi: 10.1016/j.pmcj.2022.101725.

[43] M. Chaudhary and A. Masood, "RealMalSol: real-time optimized model for Android malware detection using efficient neural networks and model quantization," *Neural Comput. Appl.*, vol. 35, no. 15, pp. 11373–11388, Feb. 2023, doi: 10.1007/s00521-023-08303-8.

[44] A. Pathak, U. Barman, and T. S. Kumar, "Machine learning approach to detect android malware using feature-selection based on feature importance score," *Journal of Engineering Research*, vol. 13, no. 2, pp. 712–720, Jun. 2025, doi: 10.1016/j.jer.2024.04.008.

[45] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Comput. Secur.*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[46] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security 2022 21:4*, vol. 21, no. 4, pp. 725–738, Feb. 2022, doi: 10.1007/s10207-022-00579-6.

[47] A. Alhussen, "Advanced Android Malware Detection through Deep Learning Optimization," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14552–14557, Jun. 2024, doi: 10.48084/etasr.7443.

[48] R. Ma, S. Yin, X. Feng, H. Zhu, and V. S. Sheng, "A lightweight deep learning-based android malware detection framework," *Expert Syst. Appl.*, vol. 255, p. 124633, Dec. 2024, doi: 10.1016/j.eswa.2024.124633.

[49] V. Sekara, L. Alessandretti, E. Mones, and H. Jonsson, "Temporal and cultural limits of privacy in smartphone app usage," *Sci. Rep.*, vol. 11, no. 1, p. 3861, Feb. 2021, doi: 10.1038/s41598-021-82294-1.

[50] A. Ehsan, C. Catal, and A. Mishra, "Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review," vol. 22, no. 20, p. 7928, Oct. 2022, doi: 10.3390/S22207928.