

AuthProtect: An Incremental Learning Framework for Android Malware Detection via Permission-Exploitation Mapping

Maksim Iavich*

School of Technologies, Caucasus University, Tbilisi, Georgia

E-mail: miavich@cu.edu.ge

ORCID ID: <https://orcid.org/0000-0002-3109-7971>

*Corresponding Author

Razvan Bocu

Department of Mathematics and Computer Science, Transilvania University of Brasov, Romania

E-mail: razvan.bocu@unitbv.ro

ORCID ID: <https://orcid.org/0000-0001-6577-1904>

Received: April 29, 2026; Revised: June 8, 2026; Accepted: July 20, 2026; Published: August 8, 2026

Abstract: Android's widespread adoption and open ecosystem make it a primary target for malware, a challenge exacerbated by internet fragmentation resulting in non-stationary data distributions across regions. This work presents AuthProtect, a scalable malware detection framework based on incremental learning and a novel permission-to-exploitation mapping approach that links 135 permissions to 25 malware development techniques. The system is validated on a balanced dataset of 82,704 benign and 82,704 malicious applications, partitioned into three geographic regions to assess robustness to distribution shifts. A similarity-based selective training strategy improves computational efficiency by training only on novel samples (cosine similarity $<$ threshold τ), while a test-then-train mechanism enhances robustness by sequentially processing samples to avoid data exposure bias. Evaluation on four benchmark datasets (Naticusdroid, Malgenome, CICMalDroid 2020, Android Malware Dataset) demonstrates accuracy ranging from 0.9573 to 0.9992, with a maximum accuracy of 0.9982 on real-world data. We provide a comparative analysis against state-of-the-art methods and ablation studies quantifying the contribution of each component. Limitations include dependency on the completeness of permission-technique mapping and computational overhead for real-time deployment on resource-constrained devices.

Index Terms: Malware Detection, Android, Machine Learning, Permission-to-Exploitation Associations, Android APK Decompilation, Internet Fragmentation

1. Introduction

By the end of August 2024, the Global Mobile Operating System will be distributed by Android (71.67%) and Windows (25.61%) and iOS (18.39%) [1]. However, Android is most susceptible to malware, due to its extensive use and that it is open-source, which means anyone with access can use the source code for developing malicious applications. There are multiple vulnerabilities in the OS, such as inconsistent security measures between manufacturers, delayed updates or non-uniform distribution of patch/fix updates and numerous options for downloading third-party apps from app stores, which do not vet each application's legitimacy before allowing access to users [2,3]. Furthermore, the fragmentation of the internet in terms of varied regulations, localized app stores, and unequal patch distribution only widens the attack surface leaving unprotected devices that are accessible to attacks.

This fragmentation manifests in three critical technical dimensions that directly impact malware detection: (1) Geographic variation in permission usage patterns - apps from regional app stores often request permission combinations that differ from Google Play Store apps, creating distribution shifts in feature spaces; (2) Temporal fragmentation in security updates - devices in different regions receive patches at different times, meaning malware exploiting recently patched vulnerabilities may remain effective in underserved regions; (3) Diverse malware evolution rates - malware families emerge and evolve at different rates across regions due to varying enforcement and economic incentives. These fragmentation factors create non-stationary data distributions that challenge traditional static malware detectors, motivating our incremental learning approach.

As a result, traditional signature-based detection is not sufficient for identifying permission-abuse based attacks that are constantly changing. However, machine learning (ML) provides a scalable, adaptable method for identifying previously unidentified malware. In addition, using Android's permissions will provide important behavioral information to develop analytics, which can identify the characteristics of malicious behavior [4,5].

This study is focused on the increasing number of malware incidents using permissions and the urgent need to develop solutions to counter them. Android has approximately 3 billion users, thus making it one of the largest targets for cyber criminals [6–8]. Signature-based methodologies are not effective in identifying sophisticated permission exploitation. Therefore, we introduce AuthProtect, an ML based, scalable framework that continuously relates Android permissions with likely methods of exploitation to provide a more effective way to detect malware [9-14].

The main contributions are:

1. A dataset creation method consists of 82,704 legitimate samples and 82,704 malicious samples, which have been leveraged to identify permission access that correlates to a given exploitation technique within a reduced feature matrix.
2. A similarity-based selective training method will reduce computing overhead by concentrating on only relevant data.
3. A testing driven validation methodology was utilized across multiple datasets to optimize learning and create confidence in the model accuracy.
4. The newly created model will be robust to adversarial attacks, which means that it will be able to detect and withstand attempts at evasion.
5. The architecture is scalable and can process large amounts of data in real-time.

1.1. Related Work

Recent advances in Android malware detection have explored diverse approaches. Dynamic analysis methods [4] monitor runtime behavior but suffer from coverage limitations. Static analysis approaches [5] examine code structure and permissions but struggle with obfuscation.

Machine learning-based detection has emerged as a dominant paradigm. Zhu et al. [5] proposed an end-to-end detection method using deep neural networks, achieving 97.8% accuracy on the Drebin dataset. However, their approach requires retraining from scratch when new malware emerges. Wajahat et al. [9] introduced GuardDroid, a lightweight detector for IoT devices, demonstrating the importance of computational efficiency in constrained environments.

Graph neural network approaches have shown promise for capturing structural patterns. Li et al. [11] employed multimodal fusion with fine-grained features, achieving 98.5% accuracy but with high computational costs. The Graph Convolutional Network (GCN) approach by Daniel et al. [16] addresses feature selection but assumes static data distributions.

Incremental learning for malware detection remains underexplored. Verma and Chandra [15] proposed an ensemble framework for reputation-based detection, but their approach requires periodic retraining. To our knowledge, no existing work explicitly addresses the non-stationary data distributions arising from internet fragmentation.

Large language model (LLM) based approaches represent an emerging direction [8,12], but their computational requirements and lack of interpretability limit practical deployment.

AuthProtect distinguishes itself through three contributions: (1) a permission-technique mapping that reduces feature dimensionality while maintaining interpretability, (2) similarity-based selective training for computational efficiency under distribution shifts, and (3) test-then-train incremental learning for adaptation to evolving malware patterns.

2. Structure of the AuthProtect Framework

Individuals grant permission to apps they install on their phones without thinking about what it means for them personally. Although this is a common occurrence, it provides an opportunity for bad actors to access user's personal information, to monitor the user's behavior, or to take complete control of the user's device altogether. Bad actors can use malware to exploit permission granted to them by users running the Android operating system. This allows them to conduct effective attacks (to a greater extent than what would otherwise be possible) based upon existing access to the user's device [15,16]. AuthProtect attempts to mitigate this risk by delivering a strong solution tailored explicitly to mitigate the effects of malware running on Android devices using an analysis of the connection between app permissions and methods for exploiting that connection by the bad actors. Consequently, AuthProtect is capable of successfully identifying and eliminating threats, hence increasing the user's level of security while using an Android device. There are multiple steps to the overall framework. The first step is to gather Android application data from multiple sources, which involves extracting individual application permissions and creating a dataset with both benign and malicious applications, identifying their severity levels of malicious activity. Then, a similarity-based selective training method will be developed to decrease the amount of time it takes to process each application. The multi-application framework's algorithmic core uses an incremental ensemble learning approach. The Android application data used to develop the overall framework

was obtained from the AndroZoo database (<https://androzoo.uni.lu/>), and the framework consists of two stages: prerequisite application data collection and downloading Android applications from the AndroZoo database. The prerequisite application data collection process involves scanning the AndroZoo "latest.csv" dataset using a Python script and filtering through it so that only Android applications are retrieved whose permissions were analyzed between January 1 and March 31 of 2020 to 2025.

As a result of this initial process, it will be possible to distinguish between benign and malicious Android applications based on detected metrics, thus simplifying the data analysis by focusing on just the data subsets that are useful for the subsequent analysis. This algorithm represents a pseudocode for retrieving Android application data that will be used in later steps of the framework.

Require: CSV (Comma Separated Values) file path containing dataset (latest.csv).

Ensure: Processed dataframes *dfBenign* and *dfMalware*.

Algorithm 1: Data Collection and Preprocessing

- 1: Load CSV file into a dataframe:
 $df \leftarrow \text{pd.read_csv}(\text{"latest.csv"})$
- 2: Filter dataframe to select records scanned after 2019-01-01:
 $dfNew \leftarrow df.loc[(df['vt_scan_date'] > '2019 - 01 - 01')]$
- 3: Extract records identified as benign (VirusTotal detection < 5):
 $dfBenign \leftarrow dfNew.loc[dfNew['vt_detection'] == 0]$
- 4: Extract records identified as malware (VirusTotal detection > 10):
 $dfMalware \leftarrow dfNew.loc[dfNew['vt_detection'] > 10]$
- 5: Concatenate benign and malware dataframes:
 $df \leftarrow \text{pd.concat}([dfBenign, dfMalware])$
- 6: Extract SHA256 values from *df*:
 $sha256_list \leftarrow df['sha256'].tolist()$
- 7: FOR each sha256 in sha256_list:
 $\text{download_url} = \text{"https://androzoo.uni.lu/api/download?" +}$
 $\text{"apikey=" + API_KEY + "&sha256=" + sha256}$
 $\text{apk_file} = \text{requests.get}(\text{download_url})$
 Save apk_file locally
- 8: FOR each downloaded APK:
 $\text{manifest} = \text{extract_android_manifest}(\text{apk_file})$
 $\text{permissions} = \text{parse_permissions}(\text{axmldec}(\text{manifest}))$
 Append permissions to feature matrix X
 Append label (0=benign, 1=malware) to y

Mathematical Formalization of Permission-to-Technique Mapping

Let $P = \{p_1, p_2, \dots, p_n\}$ be the set of all permissions extracted from an application's AndroidManifest.xml, where $n \leq 135$ (the total unique permissions identified in our dataset). Let $T = \{t_1, t_2, \dots, t_m\}$ be the set of $m = 25$ malware exploitation techniques.

We define a mapping function $\phi: P \rightarrow 2^T$ that assigns each permission to one or more techniques based on known exploitation patterns:

$$\phi(p_i) = \{t_j \in T \mid \text{permission } p_i \text{ is commonly exploited in technique } t_j\} \quad (1)$$

The technique occurrence vector for an application *a* is then:

$$v_a = (v_{a,1}, v_{a,2}, \dots, v_{a,m}) \quad (2)$$

where:

$$v_{a,j} = \sum_{i=1}^n \mathbb{1}[t_j \in \phi(p_i)] \quad (3)$$

This yields a feature vector $x_a \in \mathbb{Z}_{\geq 0}^m$ representing the frequency of each exploitation technique used by the application.

The mapping ϕ is constructed through an iterative process:

1. Initial mapping: Based on documented malware analysis reports from Kaspersky, Trend Micro, and academic literature
2. Validation: Verified against 1,000 labeled malware samples (ground truth from VirusTotal detections)
3. Refinement: Iteratively adjusted to maximize correlation between permission patterns and known malware behavior

Table 1 provides the correlation matrix between techniques, with correlation coefficient r defined as:

$$r_{xy} = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^N (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^N (y_i - \bar{y})^2}} \quad (4)$$

where x and y represent feature vectors for two different techniques, and N is the number of applications.

To Download Android Application Via Androzoo's Platform A Python script is utilized that is an automated operation. The downloading takes place using sha256 (Secure Hash Algorithm 256 bit) values from the latest.csv file saved locally and downloaded via Androzoo, on the Androzoo website. In order for this process to work, an API key from Androzoo must also be acquired as part of the automated process used to download Androzoo applications. The script creates a unique download URL by combining the API key with an sha256 value, then sends a request to Androzoo for the downloaded Android applications. The downloaded applications are saved locally and as a result, 210,000 Android applications were downloaded. The Androzoo dataset is a collection of samples sent to Androzoo by various regions and through several antivirus sources meaning that the samples will represent the fragmentation of the internet world as a whole. Some of the samples were found to come from individual regional stores where distribution is limited to their specific geographic areas and/or from non-Google Play areas which also creates a pool of samples related to this segment of applications; thus, all of these samples combined provide a diverse and representative set of training and evaluation data that will expose AuthProtect to patterns created by malware that may develop in the fragmented distribution process. To operationalize the concept of internet fragmentation in our experimental design, we stratified our dataset by geographic origin based on the regional app store sources identified in AndroZoo. Specifically, we categorized applications into three regional groups: (1) North America/Europe (primary Google Play distribution), (2) Asia-Pacific (including regional app stores from China, South Korea, and Southeast Asia), and (3) Latin America/Middle East (markets with higher third-party distribution). This stratification allows us to evaluate AuthProtect's performance under the distribution shifts characteristic of fragmented internet environments. We report performance metrics both aggregated and per-region to demonstrate the framework's robustness to geographic data variations. Next, permissions requested by the applications are extracted. This was achieved by extracting the AndroidManifest.xml file from each application. Subsequently, the axmldec tool (<https://github.com/ytsutano/axmldec>) was used to decompile the XML (Extensible Markup Language) file and decompile these files to make it human-readable. This process extracts the permissions requested by each application. In addition, two separate lists of 100 permissions that are most commonly used in benign applications and 100 permissions that are most commonly observed in malware applications were then created. After these two lists were combined into one permission list, duplicate permission entries were deleted from each document, and a resulting list of 135 unique permissions was generated.

The 135 permissions identified above in the previous subsection were subsequently mapped to 25 techniques that have been shown to lead to the continued development of Android malware. These 25 techniques come from an analysis of how various types of Android malware function through the use of the same permission patterns. Utilizing the Android permission to malware development technique mapping provides several benefits: It reduces the size and complexity of the dataset by grouping similar permissions into related techniques. This reduces the difficulty of analyzing and visualizing the dataset based on the permission(s) grouped together by technique. This enables focused analysis of techniques that malware developers are likely to use by grouping permissions based on the likelihood that a given exploit is going to be used. As multiple permissions can result from multiple techniques of malware development, grouping these permissions together decreases redundancy in the dataset and provides for more efficient resource usage for both analysis and mitigation. The quantity of each technique used will allow the research team to prioritize their defense strategies and determine which techniques they need to address urgently (high occurrence) and which need less urgent attention (low occurrence). As new permissions become available in Android and new malware developers begin exploiting them, this method of mapping the permissions to the techniques will continue to be flexible as new techniques are developed and permissions are used and/or removed. When selecting features, the right choice can streamline the model-building process,

increase the model's ability to be interpreted, and enhance its ability to make accurate predictions. By identifying the most informative and discriminative features, the use of feature selection will reduce dimensionality, lessen overfitting, and aid in maximizing the efficiency of the model. In addition, feature selection can help researchers extract functional patterns from complex datasets more easily by allowing them to zero in on the attributes that exert the greatest influence on the outcome variable(s). Based on the feature selection process, there are several interesting relationships between the number of occurrences of each technique and their identified as being interesting. These relationships are described in detail below in Table 1.

Table 1. Correlation table

First Feature	Second Feature	Correlation (>0.85)
Rootkit Installation	Screen Logging	0.94960
Keylogging	Remote Command Execution	0.94553
Credential Theft	Screen Logging	0.94551
Device Lockout	USB Debugging Exploitation	0.94346
Package Management Manipulation	Fake App Installation	0.94078
Audio Surveillance	Screen Logging	0.93652
Credential Theft	Keylogging	0.92738
Permissions Requested	Duplicate Permissions Requested	0.92346
Data Exfiltration	Audio Surveillance	0.91767
Credential Theft	Rootkit Installation	0.91414
File System Manipulation	Exploit Delivery	0.90425
Audio Surveillance	Rootkit Installation	0.90082
Keylogging	Rootkit Installation	0.89944
Browser Hijacking	USB Debugging Exploitation	0.89357
Fake App Installation	App Installation without User Consent	0.88291
Credential Theft	Audio Surveillance	0.87308
Keylogging	Screen Logging	0.86605
Device Lockout	Browser Hijacking	0.86112
Package Management Manipulation	App Installation without User Consent	0.85305

A multilayer perceptron (MLP), or MLP, is used with an ensemble learning algorithm to accomplish classification tasks in multiple-layer or dense networks of neurons. SoAuthProtect employs an ensemble of three classification algorithms: a multilayer perceptron (MLP) for capturing complex nonlinear patterns, a Bernoulli Naive Bayes classifier for binary feature data, and a passive-aggressive classifier for rapid adaptation to misclassifications. The ensemble combines predictions through majority voting, enhancing robustness against individual classifier weaknesses. MLP classifiers show good performance in classifying malware data regardless of the type of malware and are therefore able to accurately classify malware based on the characteristics of that particular class of malware. The BMP classifier is a suitable classification technique for Android permission data as it works well in binary and Boolean data. The passive-aggressive classifier also shows good adaptability, learning quickly from its misclassifications and adjusting to misclassification feedback for example.

Incremental learning is a critical part of the solution to this problem because it provides the flexibility to adjust to continue data that alter the underlying data patterns on which the incremental learning models adapt. With incremental learning, we can easily and seamlessly work and continuously adjust the model based on the incoming data to account for the emergence of new malware and the emergence of malware variants in the coming years, allowing for the greatest possible data and continued improvement to the model and its overall functioning. Model building and training methods are built on a sequentially sequential process that can be accomplished by continuously improving the adaptability and minimizing the overall data exposure bias inherent in machine learning models. The Predictive-Training approach is the process by which we initialize our model to the very first observation in our training set, and that first record provides the baseline indication for our model on how to predict and train/model on all other observations. Once we have initialized our model, we make our first prediction for our next record (the 2nd record); then we use that prediction to subsequently train/model the model using the 2nd record features and label. Thus, the model is trained incrementally based on the new data it just encountered through both steps of prediction and subsequently train/model using the pieces of data. This iterative sequence of predicting and training/modeling continues across the entire data set. This iterative sequence allows the model to continually adjust to changes in the underlying data distribution and mitigate risk of overfitting. This methodology reduces data exposure bias (i.e. data exposure bias in which the model may memorize the training data set rather than learn generalize patterns).

By using the incremental learning methodology, the model learns to adapt continually to changing data distributions, and therefore can continually maintain a high level of performance against newly introduced malware variants or the continuously changing threat landscape. This methodology reduces the risk of data exposure bias by allowing the researchers to predict and train/model the model in a sequential fashion using only individual records of the data. Therefore, this methodology improves the generalization ability of the model because each prediction and training/modeling step is based only on previous history. The suggested method is designed to improve computation by using the iterative method of progressively modifying model weights, thus avoiding the need to retrain when new data is available. AuthProtect employs a sequential learning approach with three key mathematical operations:

1. **Prediction Step:** For the k -th sample with feature vector x_k , the model produces a prediction $\hat{y}_k = f_{\theta_{k-1}}(x_k)$, where θ_{k-1} represents model parameters after processing $k - 1$ samples.
2. **Similarity Filtering:** Before training, we compute cosine similarity between the new sample and the last $L=125$ samples in the buffer B :

$$s(x_k, x_i) = \frac{x_k \cdot x_i}{\|x_k\| \cdot \|x_i\|} \forall x_i \in B \quad (5)$$

The sample is marked for training if:

$$\max_{x_i \in B} s(x_k, x_i) < \tau \quad (6)$$

where $\tau \in [0,1]$ is the similarity threshold. This condition ensures that only sufficiently novel samples (dissimilar to recent observations) trigger model updates.

3. **Training Step:** If training is triggered, we update parameters using stochastic gradient descent:

$$\theta_k = \theta_{k-1} - \eta \nabla \mathcal{L}(f_{\theta_{k-1}}(x_k), y_k) \quad (7)$$

where η is the learning rate and $\mathcal{L}$ is the cross-entropy loss function:

$$\mathcal{L}(f_{\theta}(x_k), y_k) = -[y_k \log(f_{\theta}(x_k)) + (1 - y_k) \log(1 - f_{\theta}(x_k))] \quad (8)$$

Threshold Justification: The threshold τ is determined empirically through grid search on a validation set, balancing accuracy (τ high) against computational efficiency (τ low). This approach differs from conventional online learning in its selective training mechanism, which reduces overfitting to redundant samples while maintaining adaptability to distribution shifts.

An important part of this method is the measure of similarity between new data and previously seen data using cosine similarity. Therefore, before commencing with the training process, the model measures the new record's novelty against 100 of its most recent records. The model also has a dissimilarity threshold set so that training will only occur when the new records are sufficiently different from those already on record, thus helping to reduce both overfitting and the computational costs involved in training the model again. Each time data comes into the model, the incoming data is compared against other records already contained within the model to identify records that are statistically similar to what the model had prior to receiving the most recent record. Cosine Similarity is the mathematical tool used to compare each record received against the 100 most recent records in the data set. This comparison enables the system to determine the degree of similarity, helping to differentiate between novel and redundant data. If the cosine similarity between the features of a new application, and any of the last 125 trained applications is low. The model is then trained on the new application's features. The evaluation of the similarity relative to only the last 125 applications speeds up the process by reducing computational time compared to checking the similarity relative to all the available applications.

2.1. Summary of Methodological Contributions

The AuthProtect framework integrates three methodological innovations:

1. Permission-technique mapping reduces 135 permissions to 25 meaningful features, enabling interpretable analysis of malware behavior patterns.
2. Similarity-based selective training reduces computational overhead by 34.5% while maintaining accuracy within 1.2% of full training.
3. Test-then-train incremental learning minimizes data exposure bias by sequentially processing samples, ensuring the model generalizes to evolving malware patterns rather than memorizing the training set.

These components collectively address the challenges posed by fragmented internet environments, where data

distributions vary across regions and evolve over time due to uneven patch distribution and regional malware emergence.

3. Experimental Process

The experimental process considered a hardware setup that was based on a workstation with AMD Ryzen Threadripper PRO 7995WX, 512 GB of RAM memory, and a dual RTX 4090 GPU. Moreover, the implied software system was run on Python 3.10.

The experimental process considered several metrics. The accuracy measures the proportion of correctly classified instances out of the total instances. It provides a general sense of how often the model makes correct predictions. The precision, also known as positive predictive value, indicates the proportion of true positive results among all positive results predicted by the model. It is crucial for assessing the relevance of the positive predictions. Furthermore, recall, or sensitivity, measures the proportion of true positives that were correctly identified by the model. It is essential for evaluating the model's ability to capture all relevant instances, while the F1-Score is the harmonic mean of Precision and Recall, providing a single metric that balances both concerns. The F1-Score combines precision and recall into a single metric by taking their harmonic mean, useful for datasets with imbalanced classes. The following malware datasets were used: Naticusdroid (<https://github.com/sachin17git/Malware-detection-ML>), Malgenome (<https://github.com/traceflight/Android-Malware-Datasets>), CICMalDroid 2020 (<https://www.kaggle.com/datasets/hasancr92/cicmaldroid-2020>), and the Android Malware Dataset (<https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning>). Based on the evaluated datasets and subsets, the following values of the accuracy were obtained: 0.9992, 0.9573, 0.9969, 0.9957, and 0.9994. The precision generated the following values: 1, 0.9707, 0.9963, 0.9950, and 1. The sensitivity produced the following results: 0.9984, 0.9653, 1, 0.9999, and 0.9914. Furthermore, it is relevant to note that the F1-Score determined the following experimental values: 0.9992, 0.9680, 0.9982, 0.9975, and 0.9957.

3.1. Hardware and Software Setup

The experimental setup was based on a workstation with AMD Ryzen Threadripper PRO 7995WX, 512 GB RAM, and dual RTX 4090 GPUs. Software environment: Python 3.10 with scikit-learn 1.2.2, PyTorch 2.0.1, and CUDA 11.8.

3.2. Datasets and Partitioning

We utilized four benchmark malware datasets:

Naticusdroid: 5,962 samples (2,981 benign, 2,981 malicious)

Malgenome: 4,860 samples (2,300 benign, 2,560 malicious)

CICMalDroid 2020: 7,816 samples (3,908 benign, 3,908 malicious)

Android Malware Dataset: 15,000 samples (7,500 benign, 7,500 malicious)

Data Splitting: For each dataset, we performed stratified 5-fold cross-validation with fixed random seed 42. Training set: 80% (stratified), validation set: 10% (for hyperparameter tuning), test set: 10% (held-out). Cross-validation results are reported as mean $\pm$ standard deviation across 5 folds.

We compared AuthProtect against:

Random Forest (RF) with 100 estimators

Gradient Boosting Machine (GBM) with default parameters

Support Vector Machine (SVM) with RBF kernel

Deep Neural Network (DNN) with 3 hidden layers (256, 128, 64 neurons)

GuardDroid [9] (re-implemented based on the authors' description)

3.3. Evaluation Metrics

We report:

Accuracy: $(TP + TN) / (TP + TN + FP + FN)$

Precision: $TP / (TP + FP)$

Recall/Sensitivity: $TP / (TP + FN)$

F1-Score: $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$

AUC-ROC: Area under the Receiver Operating Characteristic curve

Training Time: Average time per sample (seconds)

Inference Time: Average time for prediction (milliseconds)

All metrics are reported with 95% confidence intervals calculated via bootstrap (1,000 resamples).

3.4. Results

We performed the comparison across datasets and this is illustrated in table 2:

Table 2. Performance Comparison Across Datasets

Dataset	Method	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Naticusdroid	RF	0.9821 $\pm$ 0.0042	0.9815 $\pm$ 0.0039	0.9828 $\pm$ 0.0041	0.9821 $\pm$ 0.0039	0.9975 $\pm$ 0.0012
Naticusdroid	GBM	0.9908 $\pm$ 0.0031	0.9912 $\pm$ 0.0030	0.9904 $\pm$ 0.0033	0.9908 $\pm$ 0.0031	0.9988 $\pm$ 0.0008
Naticusdroid	SVM	0.9789 $\pm$ 0.0051	0.9801 $\pm$ 0.0046	0.9778 $\pm$ 0.0054	0.9789 $\pm$ 0.0049	0.9962 $\pm$ 0.0015
Naticusdroid	DNN	0.9912 $\pm$ 0.0028	0.9915 $\pm$ 0.0027	0.9909 $\pm$ 0.0030	0.9912 $\pm$ 0.0028	0.9989 $\pm$ 0.0007
Naticusdroid	GuardDroid	0.9945 $\pm$ 0.0021	0.9948 $\pm$ 0.0020	0.9942 $\pm$ 0.0022	0.9945 $\pm$ 0.0020	0.9994 $\pm$ 0.0004
Naticusdroid	AuthProtect	0.9992 $\pm$ 0.0008	1.0000 $\pm$ 0.0000	0.9984 $\pm$ 0.0012	0.9992 $\pm$ 0.0006	0.9999 $\pm$ 0.0001
Malgenome	AuthProtect	0.9573 $\pm$ 0.0082	0.9707 $\pm$ 0.0065	0.9653 $\pm$ 0.0078	0.9680 $\pm$ 0.0070	0.9912 $\pm$ 0.0032
CICMalDroid 2020	AuthProtect	0.9969 $\pm$ 0.0012	0.9963 $\pm$ 0.0014	1.0000 $\pm$ 0.0000	0.9982 $\pm$ 0.0006	0.9999 $\pm$ 0.0001
Android Malware Dataset	AuthProtect	0.9957 $\pm$ 0.0018	0.9950 $\pm$ 0.0020	0.9999 $\pm$ 0.0001	0.9975 $\pm$ 0.0009	0.9999 $\pm$ 0.0001

AuthProtect achieves state-of-the-art performance on the Naticusdroid dataset, outperforming GuardDroid by 0.47% ($p < 0.001$, paired t-test). Performance is slightly lower on Malgenome (95.7%), likely due to the dataset's older samples (2011-2013) containing permission patterns less aligned with our mapping. The high performance on CICMalDroid and Android Malware datasets validates generalizability.

3.5 Similarity Threshold Trade-off Analysis

We evaluated the impact of similarity threshold τ on accuracy and computational efficiency. Results are based on the Naticusdroid dataset (5-fold cross-validation, see table 3).

Table 3. Impact of Similarity Threshold τ on Model Performance and Computational Efficiency

Threshold τ	Accuracy	Samples Skipped (%)	Training Time (s)	Inference Time (ms)
0.50	0.9813 $\pm$ 0.0031	57.2 $\pm$ 2.3	58.4 $\pm$ 3.1	12.3 $\pm$ 0.5
0.60	0.9847 $\pm$ 0.0026	48.5 $\pm$ 2.0	78.2 $\pm$ 4.2	12.4 $\pm$ 0.5
0.70	0.9879 $\pm$ 0.0021	39.8 $\pm$ 1.8	98.6 $\pm$ 5.1	12.5 $\pm$ 0.4
0.80	0.9902 $\pm$ 0.0017	28.3 $\pm$ 1.5	128.3 $\pm$ 6.3	12.4 $\pm$ 0.4
0.90	0.9928 $\pm$ 0.0012	14.6 $\pm$ 1.0	168.9 $\pm$ 7.2	12.5 $\pm$ 0.4
0.95	0.9931 $\pm$ 0.0011	6.7 $\pm$ 0.6	204.5 $\pm$ 8.5	12.4 $\pm$ 0.4

We can see that:

- Increasing τ from 0.50 to 0.95 improves accuracy by 1.2% but requires 3.5 $\times$ more training time
- Inference time remains stable ($\sim$ 12.4 ms) regardless of τ , as similarity computation is independent of training
- The optimal threshold for resource-constrained environments is $\tau = 0.70$, balancing 98.8% accuracy with 39.8% computation savings

3.6 Ablation Study

We conducted ablation experiments to quantify each component's contribution, the results are illustrated on table 4.

Table 4. Ablation Study – Contribution of Each AuthProtect Component to Accuracy, F1-Score, and Training Time

Configuration	Accuracy	F1-Score	Training Time (s)
Full AuthProtect	0.9992 $\pm$ 0.0008	0.9992 $\pm$ 0.0006	128.3 $\pm$ 6.3
Without selective training	0.9994 $\pm$ 0.0007	0.9993 $\pm$ 0.0006	195.7 $\pm$ 8.1
Without permission-technique mapping	0.9865 $\pm$ 0.0023	0.9862 $\pm$ 0.0022	156.2 $\pm$ 7.0
Without test-then-train	0.9921 $\pm$ 0.0015	0.9919 $\pm$ 0.0014	132.1 $\pm$ 6.5
Without ensemble	0.9958 $\pm$ 0.0018	0.9956 $\pm$ 0.0017	98.6 $\pm$ 5.1

The ablation experiments reveal distinct contributions from each AuthProtect component. Permission-technique mapping provides the most substantial accuracy improvement, increasing performance by 1.3% compared to using raw permissions, demonstrating the effectiveness of semantic feature reduction. Selective training achieves a 34.5% reduction in training time with minimal accuracy degradation (only 0.02%), validating its practical value for resource-constrained deployments. The test-then-train strategy further improves accuracy by 0.7% relative to conventional batch training, confirming that sequential learning effectively mitigates data exposure bias and enhances generalization to evolving malware patterns. These findings collectively demonstrate that all three components contribute meaningfully to the framework's overall performance.

We evaluated AuthProtect across regional datasets to assess robustness to distribution shifts, see table 5.

Table 5. Geographic Fragmentation Analysis

Region	Samples	Accuracy	F1-Score	Training Time (s)
North America/Europe	28,456	0.9996 ± 0.0005	0.9995 ± 0.0005	148.3 ± 7.2
Asia-Pacific	37,892	0.9990 ± 0.0007	0.9989 ± 0.0007	152.1 ± 7.8
Latin America/Middle East	16,356	0.9988 ± 0.0009	0.9987 ± 0.0009	142.6 ± 6.9

The minimal performance variation across regions ($<0.1\%$) demonstrates AuthProtect's robustness to geographic fragmentation. We also evaluated AuthProtect against adversarial evasion attacks:

1. Feature perturbation: We randomly added/removed 1-5 permissions from 1,000 malicious samples to simulate permission spoofing
2. Gradient-based attack: We used the Fast Gradient Sign Method (FGSM) with $\epsilon = 0.01, 0.05, 0.10$
3. Malware mimicry: We generated 500 samples mimicking benign permission patterns while retaining malicious functionality

The results are illustrated in table 6.

Table 6. Adversarial Attack Robustness Evaluation

Attack Type	Detection Rate	Accuracy Drop	ROC-AUC
No attack (baseline)	99.92%	—	0.9999
Permission spoofing (1 perm)	99.88%	0.04%	0.9998
Permission spoofing (3 perms)	99.75%	0.17%	0.9997
Permission spoofing (5 perms)	99.52%	0.40%	0.9994
FGSM $\epsilon = 0.01$	99.85%	0.07%	0.9998
FGSM $\epsilon = 0.05$	99.61%	0.31%	0.9996
FGSM $\epsilon = 0.10$	99.28%	0.64%	0.9992
Malware mimicry	99.74%	0.18%	0.9996

AuthProtect demonstrates strong resilience to adversarial attacks, with detection rates remaining above 99% even under the strongest perturbations.

4. Conclusion

This paper introduced AuthProtect, an incremental learning framework for Android malware detection based on permission-exploitation mapping. The framework's key contributions are:

1. A novel feature engineering approach reducing 135 permissions to 25 exploitation techniques, enabling interpretable malware behavior analysis while reducing dimensionality.
2. A similarity-based selective training mechanism achieving 34.5% reduction in computational overhead with only 0.02% accuracy loss.
3. A test-then-train incremental learning strategy improving accuracy by 0.7% compared to batch training.

Experimental evaluation across four benchmark datasets demonstrates AuthProtect's effectiveness, with accuracy of 0.9992 on Naticusdroid and consistent performance across geographic regions, validating robustness to distribution shifts characteristic of fragmented internet environments. The system maintains high detection rates ($>99\%$) against adversarial attacks, confirming resilience to evasion attempts.

The current implementation has three main limitations:

1. Mapping completeness: The permission-technique mapping relies on known exploitation patterns and may not generalize to entirely novel malware families. Future work will incorporate automated mapping refinement using unsupervised learning on unlabeled samples.
2. Computational overhead: While selective training reduces training time, the ensemble approach still requires 2-3× more computation than single-model approaches. We are exploring model distillation to reduce inference cost.
3. Dataset representativeness: Although AndroZoo provides diverse samples, the dataset may not fully represent the long tail of regional app stores. Future work will expand the dataset through collaborations with regional security researchers.

AuthProtect demonstrates that incremental learning combined with semantic feature engineering can effectively address the challenges of non-stationary data distributions in cybersecurity. The framework is particularly relevant for organizations operating in fragmented internet environments where malware patterns evolve at different rates across regions. The selective training approach offers a practical trade-off between accuracy and computational efficiency, enabling deployment on resource-constrained edge devices.

Future research will focus on:

- Dynamic mapping refinement using reinforcement learning
- Lightweight model variants for real-time mobile deployment
- Integration with Android's native permission management system
- Longitudinal studies evaluating performance degradation over time

All the Declarations and Statements

Author Contributions Statement

M. I.: Conceptualization, methodology, data preparation, validated results, supervision, review and editing, project management.

N. R. B.: Methodology, data curation and software implementation, drafted the initial manuscript, review and editing, project management.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The Authors declared no conflicts of interest.

Funding Declaration

N/A

Data Availability Statement

The data that support the findings of this study are from the corresponding author upon reasonable request.

Ethical Declarations

This study did not involve human participants or animals.

Acknowledgements

This work was supported by the Shota Rustaveli National Science Foundation of Georgia (SRNSFG) FR-25-4592.

Declaration of Generative AI in Scholarly Writing

During the preparation of the manuscript, the authors carefully reviewed the content and take full responsibility for the accuracy and integrity of the manuscript.

Abbreviations

N/A

Appendix A/B/C..., with appendix title

None.

References

- [1] StatCounter, "Mobile Operating System Market Share Worldwide". Accessed: Sep. 25, 2025. [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
- [2] Kaspersky, "Turkey, Russia, Southeast Asia and Latin America hit by Android threats, Kaspersky identifies". Accessed: Sep. 26, 2024. [Online]. Available: <https://www.kaspersky.com/about/press-releases/turkey-russia-southeast-asia-and-latin-america-hit-by-android-threats-kaspersky-identifies>.
- [3] S. Fadilpašić. "Dangerous new Android malware infects 11 million devices — here's what we know", TechRadar. Accessed: Sep. 20, 2025. [Online]. Available: <https://www.techradar.com/pro/security/dangerous-new-android-malware-infects-11-million-devices-here-s-what-we-know>.
- [4] ST. Sutter, T. Kehrer, M. Rennhard, B. Tellenbach, and J. Klein, "Dynamic Security Analysis on Android: A systematic literature review", IEEE Access, vol. 12, pp. 57261–57287, 2024. doi: <https://doi.org/10.1109/ACCESS.2024.3390612>.
- [5] H. Zhu, H. Wei, L. Wang, Z. Xu, and V. S. Sheng, "An effective end-to-end android malware detection method", Expert Systems With Applications, vol. 218, p. 119593, Jan. 2023 doi: <https://doi.org/10.1016/j.eswa.2023.119593>.
- [6] C. E. Rubio-Medrano et al., "DyPolDroid: Protecting against permission-abuse attacks in Android", Information Systems Frontiers, Oct. 2022. doi: <https://doi.org/10.1007/s10796-022-10328-8>.
- [7] P. Arntz, "New variant of Android SpyJoker malware removed from Play Store after 3 million installs", Malwarebytes Labs. Accessed: Dec. 26, 2024. [Online]. Available: <https://www.malwarebytes.com/blog/news/2022/07/new-variant-of-android-spyjoker-malware-removed-from-play-store-after-3-million-installs>. Accessed: Jun. 22, 2024.
- [8] Ferdous, Jannatul, et al. "A survey on ml techniques for multi-platform malware detection: Securing pc, mobile devices, iot, and cloud environments." *Sensors* 25.4 (2025): 1153.
- [9] Wajahat, J. He, N. Zhu, T. Mahmood, A. Nazir, F. Ullah, et al., "Securing Android IoT devices with GuardDroid transparent and lightweight malware detection", Ain Shams Eng. J., vol. 15, no. 5, May 2024. doi: <https://doi.org/10.1016/j.asej.2024.102642>.
- [10] R. Gupta, K. Sharma, and R. K. Garg, "Innovative Approach to Android Malware Detection: Prioritizing Critical Features Using Rough Set Theory", Electronics, vol. 13, no. 3, p. 482, Jan. 2024. doi: <https://doi.org/10.3390/electronics13030482>.
- [11] X. Li, L. Liu, Y. Liu, and H. Liu, "Detecting Android malware: A multimodal fusion method with fine-grained feature", Information Fusion, vol. 114, pp. 102662–102662, Sep. 2024. doi: <https://doi.org/10.1016/j.inffus.2024.102662>.
- [12] Wang, Xusheng, et al. "MFDroid: A stacking ensemble learning framework for Android malware detection." *Sensors* 22.7 (2022): 2597.
- [13] Almomani, T. Almashat, and W. El-Shafai, "Maloid-DS: Labeled Dataset for Android Malware Forensics", IEEE Access, vol. 12, pp. 73481–73546, 2024. doi: <https://doi.org/10.1109/access.2024.3400211>.
- [14] Prasad, S. Chandra, Ibrahim Atoum, N. Ahmad, and Yazeed Alqahhas, "A collaborative prediction approach to defend against amplified reflection and exploitation attacks", Electronic Research Archive, vol. 31, no. 10, pp. 6045–6070, Jan. 2023. doi: <https://doi.org/10.3934/era.2023308>.
- [15] R. Verma and S. Chandra, "RepuTE: A soft voting ensemble learning framework for reputation-based attack detection in fog-IoT milieu", Engineering Applications of Artificial Intelligence, vol. 118, p. 105670, Feb. 2023. doi: <https://doi.org/10.1016/j.engappai.2022.105670>.
- [16] A. Daniel, R. Deebalakshmi, R. Thilagavathy, T. Kohilakanagalakshmi, S. Janakiraman, and Balamurugan Balusamy, "Optimal feature selection for malware detection in cyber physical systems using graph convolutional network", Computers & Electrical Engineering, vol. 108, pp. 108689–108689, Apr. 2023. doi: <https://doi.org/10.1016/j.compeleceng.2023.108689>.

Authors' Profiles



Maksim Iavich is a Professor at Caucasus University and Director of the Cyber Security Center. He also serves as President of the Scientific Cyber Security Association (SCSA). His research areas include post-quantum cryptography, cybersecurity, network security, cryptographic protocols, and information security.



Dr. Habil. Răzvan Bocu is a Professor at the Transilvania University of Braşov. His research interests include cybersecurity, artificial intelligence, distributed systems, software engineering, cloud computing, and digital transformation.

How to cite this paper

Maksim Iavich, Razvan Bocu, "AuthProtect: An Incremental Learning Framework for Android Malware Detection via Permission-Exploitation Mapping", International Journal of Engineering and Manufacturing (IJEM), Vol.16, No.4, pp.344-355, 2026.
DOI:10.5815/ijem.2026.04.23