Fake App Permissions Detection using Machine Learning

Name: G. Krishna Manohar Reddy

Roll No: 160123737040

Cyber Security Assignment - 2

1. Problem Statement

Malicious Android apps often request excessive or suspicious permissions to steal sensitive data, perform unauthorized actions, or disrupt device functionality. Detecting such apps is essential for mobile security.

The objective of this project is to build a **machine learning model** that classifies Android apps as **benign** or **malicious** based on their requested permissions using feature selection and efficient classifiers.

2. Technologies & Tools Used

- Programming : Python 3.12 + Google Colab
- Libraries : pandas, scikit-learn, matplotlib, seaborn, xgboost
- Dataset : NATICUSdroid Android Permissions Dataset (CSV, 86 features + target, ~29,333 apps)
- Features : 86 Android permissions (binary: 0 = not requested, 1 = requested)
- Label : Result column (benign → 0, malicious → 1)
- Version Control :Github

3. Research Paper

Paper:
Rathore, H., Sahay, S. K., Rajvanshi, R., & Sewak, M. (2020). Identification of Significant Permissions for Efficient Android Malware Detection. Microsoft Research.

Summary:
Analyzed Android permissions and applied feature selection methods (Variance Threshold, PCA, Autoencoders).

4. Research Gap

The paper only used static declared permissions for malware detection.
It reduced features to a small subset (~16) for efficiency.

Lightweight, permission-only detection is effective, but the study did not visualize the top permissions for interpretability.

5. Proposed Improvement

The improvement involves upgrading the model and analyzing feature contributions:

- Implemented XGBoost Classifier in addition to Random Forest.
- Selected lightweight and computationally efficient permission-based features
- Top 20 permissions selected using Variance Threshold and Chi-Square (SelectKBest).
- Visualized feature importance of the selected permissions to identify the most indicative permissions for malicious apps.
- These steps make the model faster, interpretable, and slightly more efficient compared to using all 86 permissions.

6. Methodology

Machine learning algorithms were applied to detect malicious Android apps using permissions.

Workflow:

Data Preprocessing:
Encoded Result column (benign → 0, malicious → 1).
Split dataset into X (features) and y (target).

Feature Selection:
VarianceThreshold: removed low-variance permissions.
Chi-Square (SelectKBest): selected top 20 most significant permissions.

Model Training:
Train/Test Split: 80% training, 20% testing.
Trained Random Forest and XGBoost classifiers.

Results:
Confusion matrices plotted.
Feature importance plotted for top 20 permissions.

Key Findings:
Random Forest achieved high accuracy and F1-score.
Top permissions (from feature importance) were most indicative of malicious apps.
Feature selection reduced dimensionality while retaining high performance.

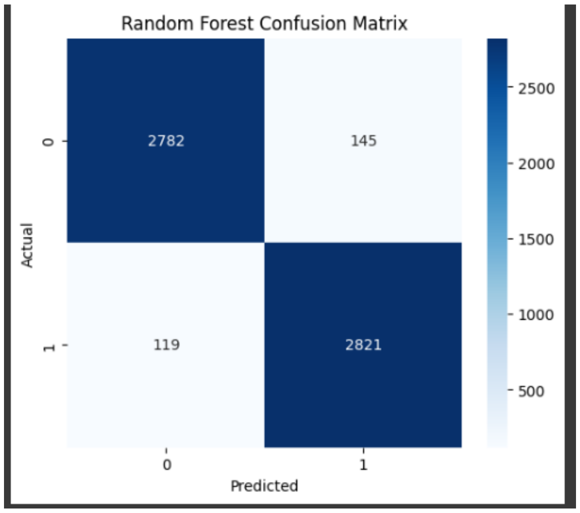
7. Results and Screenshots

Metrics Table:

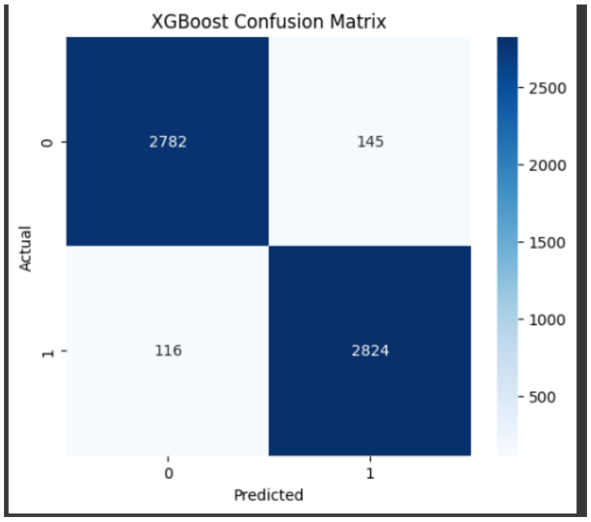
Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.95500	0.95111	0.95952	0.95529
XGBoost	0.95551	0.95116	0.960544	0.95583

Screenshots:

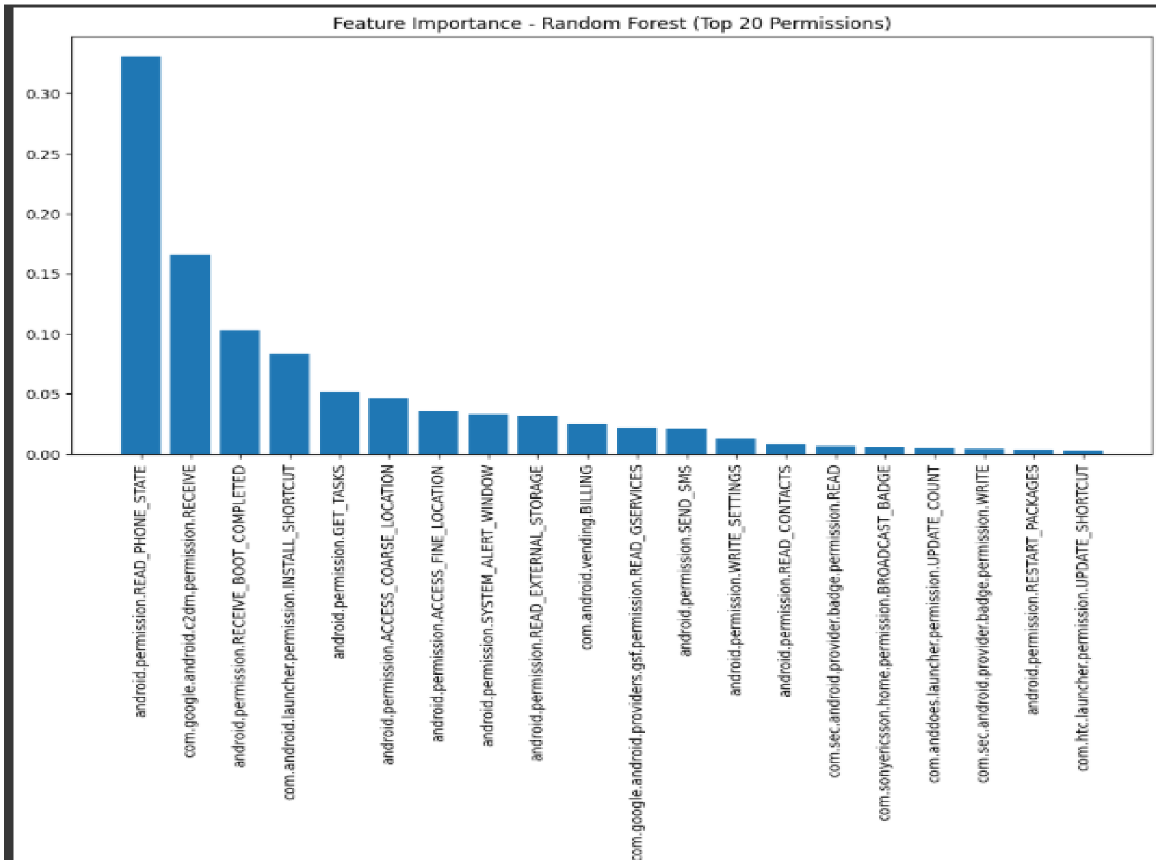
- Confusion Matrix (Random Forest)



- Confusion Matrix (XGBoost)



- Feature Importance Plot (Top 20 Permissions)



9. Conclusion

Random Forest and XGBoost successfully detected malicious apps using permissions.
Feature selection reduced dimensionality from 86 → 20 permissions while maintaining high accuracy.
Top permissions were visualized, providing interpretability for analysts.

Future Work:
- Extend to larger datasets or newer Android apps.
- Explore additional classifiers if required.

10. GitHub Repository

The complete code and dataset are available at:
https://github.com/gkmr907/Fake_App_Permissions_Detection