**Paper 1: Unraveling the Key of Machine Learning Solutions for Android Malware Detection**

**Link:** [arxiv.org/pdf/2402.02953](https://arxiv.org/pdf/2402.02953)

**Input**

* **APK Files:** The study examines Android Package Kits (APKs), which contain app code, resources, and related files.
* **Extracted Features:**
  + **Manifest Features:** Permissions, intents, application components, hardware components.
  + **DEX Files:** API call data, bytecode, opcodes, and code strings.
  + **Library Files:** Bytecode and opcodes from native libraries.
  + **Resource Files:** Static assets like images and animations.
* **Dataset:** 221,310 Android apps sourced from the AndroZoo repository (2011–2020), with malware making up around 10% of the dataset.

**Output**

* A comparison of 12 machine learning-based malware detection techniques.
* Development of **FrameDroid**, a framework for standardizing evaluations.
* Key insights:
  + More complex models, like deep learning, don’t always lead to better malware detection.
  + Adding more features doesn’t necessarily improve accuracy.
  + Malware detection struggles in real-world scenarios due to malware evolution and adversarial attacks.

**Methodology**

1. **APK Analysis:** Extract features from various APK components.
2. **Feature Representation:** Transform extracted features into different formats (categorical, image-based, text, and graph representations).
3. **Machine Learning Models:** Implement and test various models including SVM, Random Forest, CNNs, and GNNs.
4. **Framework Development:** Build **FrameDroid** to ensure consistent testing under diverse conditions.
5. **Evaluation:** Measure performance using F1-score, accuracy, and robustness under different malware scenarios.

**Paper 2: DetectBERT – A New Approach to Android Malware Detection**

**Link:** [arxiv.org/pdf/2408.16353](https://arxiv.org/pdf/2408.16353)

**Input**

* **Dataset:** 158,803 Android apps from the **DexRay dataset** (from AndroZoo), including:
  + 96,994 benign apps
  + 61,809 malware apps
* **Features:**
  + Smali code classes extracted from APKs
  + Class-level embeddings using **DexBERT**
* **Existing Models:** Uses **DexBERT** for class-level representation learning.

**Output**

* **DetectBERT**, a novel detection model that:
  + Aggregates class-level features into **app-level representations** using **correlated Multiple Instance Learning (c-MIL)**.
  + Outperforms existing malware detection models.
  + Shows strong performance in identifying new malware.
* **Public dataset and source code released** for replication.

**Methodology**

1. **Feature Extraction:** Extract Smali code from APKs and generate embeddings using **DexBERT**.
2. **Instance Aggregation:** Treat each Smali class as an instance within an **MIL bag**, utilizing Nyström Attention for feature correlation.
3. **Feature Fusion:** Merge class embeddings into a comprehensive **app-level representation**.
4. **Model Training:** Train the model on an 80/10/10 split for training, validation, and testing.
5. **Evaluation:** Assess the model's accuracy, precision, recall, and F1-score, including performance in detecting new malware variants.

**Paper 3: LensDroid – Visualizing Android App Behaviors for Malware Detection**

**Link:** [arxiv.org/pdf/2410.06157](https://arxiv.org/pdf/2410.06157)

**Input**

* **Dataset:** 51,165 apps from **AndroZoo** and **Drebin datasets**.
  + 25,733 benign apps, 25,432 malware samples.
* **Features Visualized:**
  + **Behavioral Sensitivities:** API call graphs.
  + **Operational Contexts:** Opcode-based matrices from Smali code.
  + **Supported Environments:** Binary-transformed images from .dex, .xml, and .so files.

**Output**

* **LensDroid**, a multi-view malware detection system that:
  + Outperforms five baseline detection methods.
  + Excels in identifying **zero-day threats** and **app evolution scenarios**.
  + Demonstrates the **effectiveness of multi-view feature fusion**.

**Methodology**

1. **Feature Visualization:** Convert app behaviors into **three visual representations** (API graphs, opcode matrices, binary images).
2. **Feature Extraction:**
   * **GCN** for API call graphs
   * **TextCNN** for opcode matrices
   * **ConvNeXt** for binary-transformed images
3. **Multi-View Fusion:** Integrate extracted features using **Multi-modal Factorized Bilinear (MFB) pooling** and **multi-head self-attention**.
4. **Classification:** Train a **Deep Neural Network (DNN)** classifier.
5. **Evaluation:** Test model accuracy, precision, recall, and F1-score against existing approaches.

**Paper 4: Metadata-Based Malware Detection on Android Using Machine Learning**

**Link:** [arxiv.org/pdf/2307.08547](https://arxiv.org/pdf/2307.08547)

**Input**

* **Dataset:** 2.8 million Android apps categorized as:
  + 1,447,566 benign
  + 1,397,986 malware
* **Metadata Features:**
  + **App permissions** as primary detection criteria.
* **Data Sources:**
  + VirusTotal
  + AndroZoo
  + CCCS-CIC-AndMal-2020
  + Contagio Mobile mini dump

**Output**

* **A trained neural network model** for malware detection based on permissions.
* Achieved:
  + **92.93% accuracy**
  + **91.57% recall**
  + **94.13% precision**
* Fully connected neural networks (dense layers) were found to be the most effective.

**Methodology**

1. **Dataset Collection:** Gather metadata from various sources and label apps as benign or malicious.
2. **Feature Selection:**
   * Use only **permissions** for detection.
   * Filter out rare permissions (less than 26 occurrences).
   * Convert dataset into **binary vectors** of 2,138 features.
3. **Model Selection:** Compare different ML models:
   * Fully connected neural networks (best performer)
   * GRU, LSTM, and CNNs
4. **Training & Validation:**
   * Test on **Android Permissions Dataset (2019)**.
   * Train on the **full dataset** after selecting the best model.
5. **Final Evaluation:**
   * Model achieves 92.93% accuracy.
   * Future improvements could involve **adding certificate ownership features**.

**Comparison of Methods**

| **Feature** | **FrameDroid** | **DetectBERT** | **LensDroid** | **Metadata-Based Model** |
| --- | --- | --- | --- | --- |
| **Key Focus** | Evaluating ML models | App-level feature learning | Multi-view visualization | Permission-based classification |
| **Best Feature** | Standardized testing framework | Strong performance on unseen malware | Excels in zero-day detection | High efficiency & accuracy |
| **Weakness** | Struggles with malware evolution | Relies on pre-trained embeddings | Computationally expensive | Limited to permission-based features |