Multimodal and Explainable Android Adware Classification Techniques

Momoreoluwa Ayinde, Sheikh Rabiul Islam, Iman Dehzangi, Fahmida Tasnim Lisa
Rutgers University - Camden
momoreoluwa.ayinde@rutgers.edu, sheikh.islam@rutgers.edu, i.dehzangi@rutgers.edu, fahmidatasnim.lisa@rutgers.edu

Abstract

With the widespread availability of adware masquerading as useful apps, there is an increasing need for robust security measures to identify adware. The identification of adware as a malware is a challenging task, as it often appears benign despite its malicious intent in the background. In this study, we propose an approach to classify adware on Android devices using data from multiple modalities. The focus is particularly on the classification of Airpush and Dowgin adware. Our proposed method uses both tabular and grayscale image data, and a feedforward neural network architecture to build a multimodal deep learning model that achieves a 95% prediction accuracy. Additionally, we incorporate Explainable AI (XAI) to enhance the interpretability of the results. The efficiency of our proposed approach is showcased through its ability to classify adware instances in an explainable manner, highlighting its significance not only in adware classification but also in fortifying against the evolving challenges posed by adware.

Introduction

Adware, often disguised as harmless software, poses a significant threat by infiltrating popular apps and compromising user privacy and security. With over 200 widely-used Android applications containing adware, millions of users are affected globally. Addressing this multifaceted issue is crucial to safeguarding users and maintaining the digital ecosystem integrity as mobile devices become increasingly integrated into our daily life. Studies on Android malware detection using machine learning approaches can be broadly categorized into two methodologies: static analysis and dynamic analysis. In static analysis, researchers extract detailed information from the Android Package Kit (APK) installation file. This includes insights into the app's manifest, permissions, Application Programming Interface (API) calls, intents, and more. On the contrary, dynamic analysis focuses on monitoring an application's behavior during execution, examining aspects such as logcat errors, shared memory usage, system calls, and process activity within a controlled environment or sandbox. Previous studies proposed several Android malware detection methods using deep learning and ensemble learning (Deldar and Abadi 2023) (Summaira et al. 2021). Malware data can be presented in terms of different modalities such as text, code, images, and tabular data. This multifaceted nature of malware data requires an approach that can handle and process various modalities efficiently. There has been various studies using multimodal learning methods on different modalities of data. For instance, (Kwak, Jung, and Lee 2023) works with text, images, and code and uses multimodal deep learning for issue report classification. Similarly, (Summaira et al. 2021) uses multimodal deep learning and ensemble learning in their work.

Our study introduces a pioneering multimodal approach for adware classification, merging tabular data with grayscale image data. This integration captures both functional aspects and visual representations of apps in memory. Multimodal methods, combining tabular and image data, provide better predictions by capturing both behavior and visual patterns. However, they require more complex setups and resources. On the other hand, simpler unimodal methods need fewer resources and are suitable for limited setups. Despite challenges, we opted for the more comprehensive multimodal approach, though further research is necessary to fully leverage its potentials.Our approach achieves a 95% prediction accuracy in adware classification. Moreover, we go beyond classification by incorporating explainable AI to enhance result interpretability. This fosters a deeper understanding of the classification process for future research and empowers end-users with insights into flagged applications. Our contributions can be summarized as follows:

- Integration of data from multiple sources: tabular adware data from the CIC-AndMal-2020 (Keyes et al. 2021) (Rahali et al. 2020) and grayscale image adware data from (Iadarola et al. 2021)
- Evaluation of the multimodal model's effectiveness through comparison with unimodal image and tabular data models.
- Implementation of explainable AI techniques to elucidate the classification process and provide insights into underlying threats.

Related Works

(Liu et al. 2020) offers a comprehensive overview of machine learning-based malware detection methods for Android platforms, covering static, dynamic, and hybrid analysis techniques. Static analysis examines app characteristics like permissions, code structure, and API calls (Lou et al. 2019). However, adware is always evolving. Sophisticated variants dynamically inject malicious code or alter behavior at runtime, rendering static analysis blind to their true nature (Gao et al. 2019). To overcome this limitation, later studies have turned their focus to dynamic data. Dynamic analysis occurs when the app is running (Yerima, Alzaylaee, and Sezer 2019) (Ndagi and Alhassan 2019), employing behavioral analysis, and monitoring runtime app actions to identify adware-related behavior. (Kumar et al. 2019) delves into the realm of network traffic generated by adware by integrating diverse data modalities. Although their focus was broader, the foundational principles of multimodal learning method that they proposed is important to the challenges posed by adware. By identifying unique signatures and communication patterns, they demonstrated the potential of network-based detection in distinguishing malware from benign apps. Recognizing the strengths of both approaches, (Lu et al. 2020) developed a hybrid model, combining static features with dynamic data like API calls, network traffic, and system calls. Their promising results illustrated the effectiveness of harnessing both static and dynamic analysis. There has been several studies on multimodal learning using deep learning for malware detection.

In (Deldar and Abadi 2023), the authors propose a taxonomy categorizing zero-day resistant, deep malware detection, and classification techniques into unsupervised, semi-supervised, few-shot, and adversarial resistant categories. In a different study, (Zhang, Zhao, and Wang 2021) introduces SusTriage, a bug report triage method that combines multimodal deep learning and ensemble learning to achieve high prediction performance and at the same time maintain long-term sustainability of open source communities. Later on, (Summaira et al. 2021) explored the importance of multimodal deep learning to improve upon information processing by integrating various modalities such as image, video, text, audio, body gestures, facial expressions, and physiological signals.

More recently, (Kwak, Jung, and Lee 2023) proposed a novel multimodal model incorporating text, images, and code as its modalities and achieved better performance compared to its prior studies. They used a text-based Convolutional Neural Network (CNN) unimodal model, which were then fused together to enhance issue classification accuracy. In a similar study, (Audebert et al. 2020) used a multimodal neural network integrating image and word embeddings extracted from OCR-generated noisy text. Their proposed approach significantly improved classification accuracy on the Tobacco3482 and RVL-CDIP datasets. In a different study, (Iadarola et al. 2021) provided a valuable framework for designing interpretable models in the context of mobile malware detection.

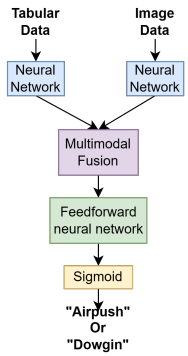

Figure 1: Proposed architecture

Methodology

Overview

Our proposed approach (see Fig.1) uses multimodal deep learning on image and tabular dataset. Additionally, we implemented explainable AI techniques to help understand decisions depicted by the most contributing features.We evaluated the unimodal tabular data on eight distinct machine-learning algorithms namely, K-Nearest Neighbors (KNN), Decision Trees, Logistic Regression, Naive Bayes, Random Forest, Extreme Gradient Boosting (XGBoost), Support Vector Machine (SVM) with a linear kernel, and SVM with a Radial Basis Function (RBF) kernel. Similarly, we evaluated the image data on a custom CNN Architecture . For the fusion of the data, we used two separate neural network models and then concatenated the output layers. Afterwards, we trained the concatenated features on a feedforward neural network model with fully-connected layers.For interpretability of the unimodal tabular model, we use Shapley Additive Explanations (SHAP) to display feature importance. We also implemented Gradient-weighted Class Activation Mapping (GradCAM) to understand how the unimodal image model classifies grayscale adware images.

Dataset

We utilized public datasets of different modalities (image and tabular) to train and evaluate our adware classification model. The employed datasets include CCCS-CIC-AndMal-2020 (Keyes et al. 2021) (Rahali et al. 2020) and Android grayscale image dataset collected in (Iadarola et al. 2021).The CCCS-CIC-AndMal-2020 dataset, publicly released in 2020, is a collaborative effort of the Canadian Centre for Cyber Security and the Canadian Institute for Cybersecurity. Here we use the malware category- Adware. We specifically leverage features extracted from the Airpush and Dowgin families, known for their adware characteristics. Building upon the research presented in (Iadarola et al. 2021), we incorporate and processed image data in this research. They are Android malware grayscale images of Airpush and Dowgin malware families. We utilized two separate datasets due to the lack of a single, cohesive dataset of both types of data. We acknowledge that this is a limitation of our study, as combining data from different sources can

introduce inconsistencies. To mitigate this, we ensured that both data sources were aligned on their labels.

Feature Extraction

The tabular data contained a total of 2927 samples which were split as: 2017 samples in the Airpush class and 910 samples in Dowgin class. We excluded the 'Hash' and 'Category' columns from the datasets due to their lack of relevance to the model. We used Scikit-Learn's Standard Scaler to standardize the tabular data. Note that there are no missing values in this dataset. Finally, we applied a OneHotEncoder on our label as a standard data pre-processing technique. For the image data, given that the image sizes are variable, we standardized the image dataset by resizing each image. The input images were resized to dimensions (258 x 258). Subsequently, a normalization process was applied, and the images were fed into a CNN based model. For Feature Extraction from the CIC-AndMal-2020 tabular dataset (Keyes et al. 2021) (Rahali et al. 2020), we used Recursive Feature Elimination (RFE) along with GridSearch. RFE is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. We performed RFE and GridSearch with four different classifiers: Random Forest, Decision Tree, Logistic Regression, and Adaptive Boosting (Adaboost). These classifiers provide a good mix of different types of models, from tree-based models (Random Forest and Decision Tree) to linear models (Logistic Regression) and ensemble methods (Adaboost). For each classifier, we defined a set of hyperparameters to be tuned using GridSearch. GridSearch is a method that performs hyperparameter tuning in an exhaustive manner, searching over all possible combinations of the provided hyperparameters. After performing RFE and GridSearch, we obtained a list of 45 common features across all classifiers.

Models

UniModal Model: Following feature extraction of the tabular dataset, we used the 8 different classifiers listed earlier to train our model.The results are presented in Table 1 in terms of accuracy, recall, precision and f1 score. For the image data, we used a custom CNN architecture on the preprocessed image data (Iadarola et al. 2021). The CNN architecture consisted of three convolutional layers with a kernel size of (5, 5) followed by the ReLU activation function. We applied a max-pooling operation twice with a kernel size of (5, 5). After the final max-pooling layer, the output is passed to a fully connected layer. Finally, it extracts the image features in a feature size consistent with the feature size of the tabular data. We used a softmax activation function for the output layer. This model was trained for 10 epochs with a batch size of 32. As a result, we achieved a 97.6% prediction accuracy on the training set and 95.3% prediction accuracy on the test set. We achieved a 97.79%, 97.9%, and 97.8% on precision, recall and F-1 score respectively.The training loss and accuracy, as well as the validation loss and accuracy for each epoch, were recorded. The training loss provides an indication of how well the model is learning from the training data, while the validation loss gives an insight into how well

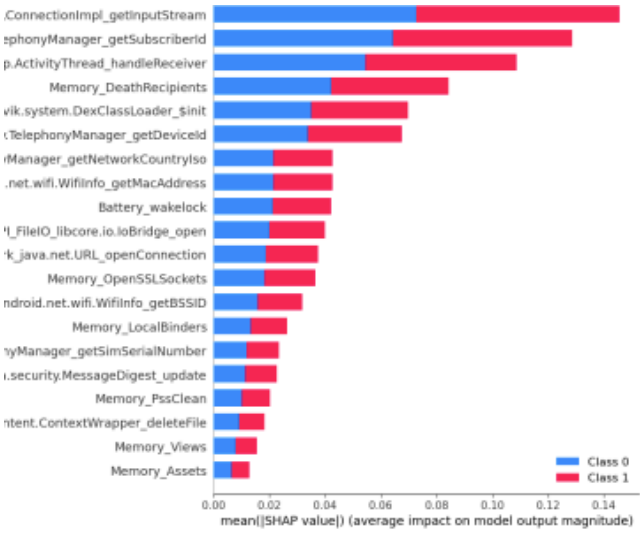

Figure 2: SHAP values of the most important 20 features of the CIC-AndMal-2020 pipeline

the model generalizes to unseen data. Ideally, both losses should decrease over time, indicating that the model is learning effectively. In our case, the training loss decreased from 0.6832 in the first epoch to 0.2157 in the tenth epoch. Similarly, the validation loss decreased from 0.6759 in the first epoch to 0.2047 in the tenth epoch.

Multimodal Model: For preprocessing, we sorted both the image and tabular data based on their labels to ensure consistency. Due to the presence of a larger amount of image data compared to the tabular data, we took a random subset of the training and test data so we have an equal amount for our model. The images were resized to a standard size of 128 x 128. The multimodal model has two neural network branches. The tabular data was processed through a dense layer with 32 units and a ReLU activation function. The image data was processed through a convolutional layer with 16 filters of size 3x3 and a ReLU activation function, followed by a flattening layer. The outputs of these two branches were then concatenated and passed through a common dense layer with 64 units and a ReLU activation function. The final output layer uses a sigmoid activation function for binary classification. The model was compiled with the Adam optimizer and binary cross-entropy loss function. The batch size was set to 16 and the model was trained for 10 epochs. These hyperparameters were chosen based on common practices in the field. However, we acknowledge that further hyperparameter tuning could potentially improve the model's performance. The same preprocessing applied to the training data was also applied to the test data.

Results and Discussion

For the unimodal tabular data, we used SHAP for explainability and from Figure 2 we see that the features: *API_Network_-com.android.okhttp.internal.huc.HttpURLConnectionImpl_-*

getInputStream, *API_DeviceInfo_android.telephony.TelephonyManager_getSubscriberId*, and *API_Binder_android.app.ActivityThread_handleReceiver* had a major impact on the classification results. *API_Network_com.android.okhttp.internal.huc.HttpURLConnectionImpl_getInputStream* represents the use of an API to retrieve data from a URL. Adware often needs to connect to the internet to download advertisements or send user data to a server. Therefore, the frequent use of this API could be a strong indicator of adware. *API_DeviceInfo_android.telephony.TelephonyManager_getSubscriberId* represents the retrieval of the subscriber ID, which is a unique identifier for each user in the mobile network. Adware often collects personal information for malicious purposes, so the use of this API could suggest an attempt to gather sensitive user data. *API_Binder_android.app.ActivityThread_handleReceiver* represents the handling of broadcast receivers in Android, which are used to respond to system-wide broadcast announcements. Adware might use this to trigger actions upon certain events, such as network changes or system boot. Among the tabular models, Random Forest and XGBoost performed the best both with 98% prediction accuracies while Naive Bayes performed the worst with an 89% prediction accuracy. Applying GradCAM to our image model provides insights into how the model learns its classification process. In Figure 3, we present the visualization of a convolutional layer depicting an image labeled "Airpush" through a heatmap and GradCAM image. Despite the inherent difficulty in distinguishing grayscale images, our comparison of the heatmaps and GradCAM images pertaining to the Dowgin adware family reveals the efficacy of GradCAM in discerning subtle distinctions. Specifically, as we traverse through various convolutional layers, the GradCAM consistently delineates the grayscale images with greater clarity, particularly in the grey-on-black instances. This observation suggests that our model learns in an interpretable manner, effectively highlighting the characteristics of the Airpush grayscale image. We evaluated our multimodal deep learning model by comparing it with unimodal model for image and unimodal model for tabular dataset. For the unimodal image model, we applied a CNN architecture and achieve 97.6% accuracy which is 2.6% higher than our multimodal model, where it achieves 95% accuracy. The multimodal model's performance was evaluated using both training loss and accuracy. Ideally, the loss should decrease over time, indicating that the model is learning effectively. In our case, the training loss decreased from 5.6638 in the first epoch to 0.6295 in the twentieth epoch. We chose to distinguish between two families of adware, Airpush and Dowgin because different families of adware exhibit different behaviors and characteristics. By distinguishing between different families, we can gain more understanding of the threats posed by each family and develop more targeted mitigation strategies.Airpush and Dowgin were chosen as the adware samples because they represent two common and distinct types of adware. Airpush is known for aggressive advertising tactics and has been associated with unwanted pop-up ads, while Dowgin is known for being bundled with seemingly legitimate apps and can be harder to detect.

Classifier	Class	Precision	Recall	F1 Score	Accuracy
XGBoost	Airpush	98%	99%	99%	98%
	Dowgin	98%	95%	97%	
Decision Tree	Airpush	96%	97%	96%	95%
	Dowgin	92%	90%	91%	
Logistic Regression	Airpush	96%	96%	96%	94%
	Dowgin	90%	92%	91%	
KNN	Airpush	96%	99%	97%	96%
	Dowgin	98%	90%	94%	
SVM with RBF Kernel	Airpush	95%	97%	96%	95%
	Dowgin	92%	89%	91%	
SVM with Linear Kernel	Airpush	95%	95%	95%	94%
	Dowgin	89%	89%	89%	
Random Forest	Airpush	98%	99%	99%	98%
	Dowgin	98%	96%	97%	
Naive Bayes	Airpush	88%	97%	92%	89%
	Dowgin	91%	71%	79%	

Table 1: Classification Matrix for Machine Learning classifiers on CIC-AndMal-2020 tabular dataset

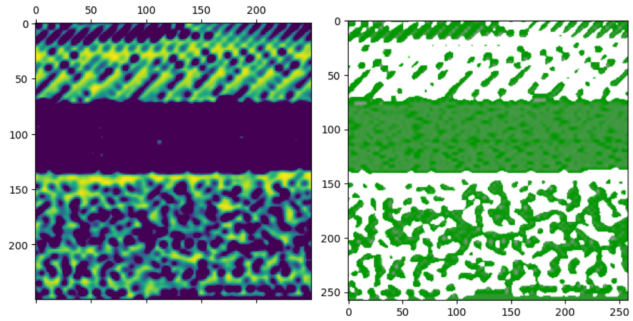

Figure 3: GradCAM on convolutional layer of Image data model. Left: Heatmap. Right: GradCAM image

Conclusion

Our multimodal deep learning approach presents a promising option for adware classification, using the complementary strengths of image and tabular data. This approach allows the model to learn more complex patterns that might not be apparent when using either type of data alone. The combination of neural networks and thorough data preprocessing contributed to the model's performance and robustness. Due to the inherent differences between the two data sources, future research could benefit from the development and use of a unified dataset that integrates tabular and image data from a single source. Future work could also investigate the benefits of distinguishing between adware and non-adware, which could provide a broader perspective on the threat landscape.

References

Audebert, N.; Herold, C.; Slimani, K.; and Vidal, C. 2020. Multimodal deep networks for text and image-based document classification. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, 427–443. Springer.

Deldar, F., and Abadi, M. 2023. Deep learning for zero-day malware detection and classification: A survey. *ACM Computing Surveys*.

Gao, J.; Li, L.; Kong, P.; Bissyandé, T. F.; and Klein, J. 2019. Should you consider adware as malware in your study? In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 604–608. IEEE.

Iadarola, G.; Martinelli, F.; Mercaldo, F.; and Santone, A. 2021. Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security* 105:102198.

Keyes, D. S.; Li, B.; Kaur, G.; Lashkari, A. H.; Gagnon, F.; and Massicotte, F. 2021. Entroplyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics. In *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 1–12. IEEE.

Kumar, R.; Zhang, X.; Wang, W.; Khan, R. U.; Kumar, J.; and Sharif, A. 2019. A multimodal malware detection technique for android iot devices using various features. *IEEE access* 7:64411–64430.

Kwak, C.; Jung, P.; and Lee, S. 2023. A multimodal deep learning model using text, image, and code data for improving issue classification tasks. *Applied Sciences* 13(16):9456.

Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; and Liu, H. 2020. A review of android malware detection approaches based on machine learning. *IEEE Access* 8:124579–124607.

Lou, S.; Cheng, S.; Huang, J.; and Jiang, F. 2019. Tfdroid: Android malware detection by topics and sensitive data flows using machine learning techniques. In *2019 IEEE 2Nd international conference on information and computer technologies (ICICT)*, 30–36. IEEE.

Lu, T.; Du, Y.; Ouyang, L.; Chen, Q.; and Wang, X. 2020. Android malware detection based on a hybrid deep learning model. *Security and Communication Networks* 2020:1–11.

Ndagi, J. Y., and Alhassan, J. K. 2019. Machine learning classification algorithms for adware in android devices: a comparative evaluation and analysis. In *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*, 1–6. IEEE.

Rahali, A.; Lashkari, A. H.; Kaur, G.; Taheri, L.; Gagnon, F.; and Massicotte, F. 2020. Didroid: Android malware classification and characterization using deep image learning. In *2020 The 10th international conference on communication and network security*, 70–82.

Summaira, J.; Li, X.; Shoib, A. M.; Li, S.; and Abdul, J. 2021. Recent advances and trends in multimodal deep learning: a review. *arXiv preprint arXiv:2105.11087*.

Yerima, S. Y.; Alzaylaee, M. K.; and Sezer, S. 2019. Machine learning-based dynamic analysis of android apps with improved code coverage. *EURASIP Journal on Information Security* 2019(1):1–24.

Zhang, W.; Zhao, J.; and Wang, S. 2021. Sustriage: Sustainable bug triage with multi-modal ensemble learning. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 441–448.