Explaining the Effectiveness of Machine Learning in Malware Detection: Insights from Explainable AI

Hendrio Bragança[1], **Vanderson Rocha**[1], **Eduardo Souto**[1], **Diego Kreutz**[2], **Eduardo Feitosa**[1]

[1]Universidade Federal do Amazonas (UFAM)
[2] Universidade Federal do Pampa (UNIPAMPA)

{hendrio.luis, vanderson, esouto, efeitosa}@icomp.ufam.br

diegokreutz@unipampa.edu.br

Abstract. *We use Explainable Artificial Intelligence (XAI) to understand and assess the decisions made by ML models in Android malware detection. To evaluate malware detection, we conducted experiments using seven datasets. Our findings indicate that it is possible to accurately identify malware across multiple datasets. However, each dataset may have a different collection of features available. We also discuss the implications of incorporating expert-dependent features into the malware detection procedure. Such features have the potential to increase model accuracy by detecting minor indicators of harmful behaviour that automated algorithms may miss. However, because of the necessity for in-depth manual analysis, this strategy increases the resource and time requirements. It also risks adding human bias into the models and raises scaling issues in the continuously developing Android application landscape. Our results suggest that XAI techniques should be used to help malware analysis researchers understand how ML models work, rather than only concentrating on increasing accuracy.*

1. Introduction

The widespread use of Android-powered devices in our daily lives has increased malicious applications, posing a threat to users' security and privacy [Aboaoja et al., 2022, Miranda et al., 2022]. In the pursuit of mitigating this danger, researchers have proposed various approaches to detect malware on Android devices, where many of them are based on machine learning (ML) techniques [Zakeya et al., 2022, Ullah et al., 2022, Talbi et al., 2022, Alani and Awad, 2022, Scalas et al., 2021].

While the availability of these techniques has improved, for example, the large-scale Android virus detection, a significant challenge lies in the "black-box" nature of these methods. The opacity often renders their decision-making processes unclear and difficult to explain. The lack of transparency, also known as the explainability problem, hampers our ability to comprehend why an application is classified as either malware or benign [Mendes and Rios, 2023].

Addressing this issue requires a deeper investigation into the decision-making processes of ML models. The degree to which decision-makers can understand and trust the predictions of these models will determine their widespread adoption in critical domains such as cybersecurity, criminal justice, and healthcare [Kinkead et al., 2021, Charmet et al., 2022]. Explainable Artificial Intelligence (XAI) techniques offer valuable

solutions in this context. These techniques shed light on the most influential features utilized by ML models and provide insights into how changes in those features can impact the models' predictions [Yin et al., 2019, Alikhademi et al., 2018]. By incorporating XAI techniques into the realm of Android malware detection, our objective is to enhance explainability while maintaining high performance [Charmet et al., 2022].

In this work we explore XAI techniques that can be applied to Android malware detection models, aiming to understand better how these models make decisions. We use the SHAP framework [Lundberg and Lee, 2017] to investigate what ML-based models learn during the training process, analyze their findings, and discuss the factors contributing to the exceptional performance of malware detection models in experimental scenarios.

To ensure comprehensive analysis, our study incorporates a variety of openly and accessible Android malware datasets, including Drebin-215 [Yerima and Sezer, 2019], Androcrawl [Sisto, 2013], KronoDroid [Guerra-Manzanares et al., 2021], Android-Permissions [Sarma et al., 2012], Adroit [Martín et al., 2016], DefenseDroid [Colaco et al., 2021], and MH-100K [Bragança et al., 2023]. These datasets offer various features for Android malware detection, representing different aspects of the Android operating system, such as permissions, system calls, hardware accesses, and API calls. This diversity of features is essential for building robust malware detection systems.

We discover that the lack of a direct intersection of feature sets across the different datasets used in this study demonstrates the wide range of Android malware operations, which requires a broader range of features for accurate detection. This diversity emphasizes the importance of utilizing multiple datasets and features to enable precise and reliable malware identification. Furthermore, we discuss the implications of incorporating expert-dependent features into the malware detection process. These features can enhance model accuracy by detecting subtle indicators of malicious behavior that ML algorithms may overlook.

In summary, our contribution is twofold. First, we provide a comprehensive XAI-based examination of feature importance in malware detection models. Our findings provide insights into the critical variables that AI/ML models consider when distinguishing between benign and malicious apps, thereby enhancing the models' knowledge and interpretability. Second, XAI Integration with malware detection models, which is a crucial to foster high-quality and advanced research. With such integration, we aim to achieve high accuracy while elucidating the underlying decision-making processes of these models, focusing on model interpretability and understanding. This approach adds validation, fostering increased trust in the models' predictions and leading to more effective malware detection mechanisms.

The remainder of this paper is organized as follows. In Section 2 we present the related works. Next, we describe the XAI methodology used in this study in Section 3. Finally, in Sections 4 and 6 we introduce our experimental protocol and results and the final remarks.

2. Background and Related Works

The detection of Android malware plays a critical role in safeguarding users of application markets and improving the scrutiny processes employed by these markets. This task involves the analysis of malware samples, requiring extracting relevant features to generate signatures or behavioral profiles. Several works propose machine learning-based approaches for effective malware detection [Odusami et al., 2018, Kouliaridis et al., 2020, Liu et al., 2020, Muzaffar et al., 2022, Bhat et al., 2023], highlighting the crucial importance of high-quality features in constructing robust detection models.

Different types of features (e.g., permissions, system calls, network traffic) have been used in ML-based malware detection research [Qamar et al., 2019, Pimenta et al., 2023]. Researchers took also advantage of additional metadata, such application description, developer ID, and application category, to improve the accuracy and effectiveness of malware detection solutions.

Regardless of the specific feature types, metadata or methods used in the analysis, the primary goal is always to improve the output metrics (e.g., accuracy, recall) of the Android malware classifier. However, it is worth emphasizing that researchers have only started focusing on understanding the decisions made by classifiers using explainability techniques in recent years. These techniques provide insights into the reasoning behind classifier predictions, adding a layer of transparency and interpretability to the malware detection process. Most publications in the literature include ML models for Android malware detection, but very few have made an effort to explain the decisions made by the models.

[Mathews, 2019] utilized the LIME (Local Interpretable Model-Agnostic Explanations) algorithm to classify malware and provided a general explanation of explainable artificial intelligence (XAI), referencing important principles for evaluating explainability. [Nellaivadivelu et al., 2020] conducted a black-box study of the Android malware system, analyzing which features a classification model relies on for making decisions. [Fan et al., 2020] evaluated five distinct local and model-agnostic explanation approaches for Android malware analysis - LIME, Anchor, LORE (LOcal Rule-based Explanations) SHAP (Shapley Additive Explanations) , and LEMNA (Local Explanation Method using Nonlinear Approximation) [Guo et al., 2018]). The authors evaluated the stability, robustness, and effectiveness of model-agnostic explanation approaches on a variety of malware classifiers, including multilayer perceptron (MLP), random forest (RF), and support vector machines (SVM).

[Kim et al., 2021] investigated the use of XAI in cybersecurity technologies to improve the efficiency of analysts' decision-making. SHAP and FOS (Feature Outlier Score) algorithms are used in the paper to uncover relevant information in IDS and malware datasets. [Melis et al., 2022] utilized gradient-based attribution approaches to explain Android malware classification decisions and selected the most relevant features. They also presented measures to evaluate the influence of explanations on the classifiers' adversarial robustness.

[Alani and Awad, 2022] proposed a lightweight Android malware detection approach that uses explainable machine learning to distinguish between harmful and benign applications. The author's results indicate an accuracy of over 98% while preserving a

minimal footprint on the device. Furthermore, Shapley Additive Explanation (SHAP) values are used to explain the classifier model.

In comparison to these existing works, our research offers a more comprehensive and in-depth examination of malware detection, with an emphasis on both accuracy and an understanding of how ML models operate utilizing XAI methodologies. It does more than just list these features; it provides an extended analysis of various datasets with various feature sets, emphasizing the diversity of Android malware operations.

3. Insights from Explainable AI

In this section, we present a two-step process for evaluating malware detection algorithms, as depicted in Figure 1. The first step follows the conventional approach commonly found in machine learning research, encompassing data source information, validation methods, malware detection model development, and evaluation. The second step introduces the incorporation of recent explainability techniques into the traditional ML pipeline.

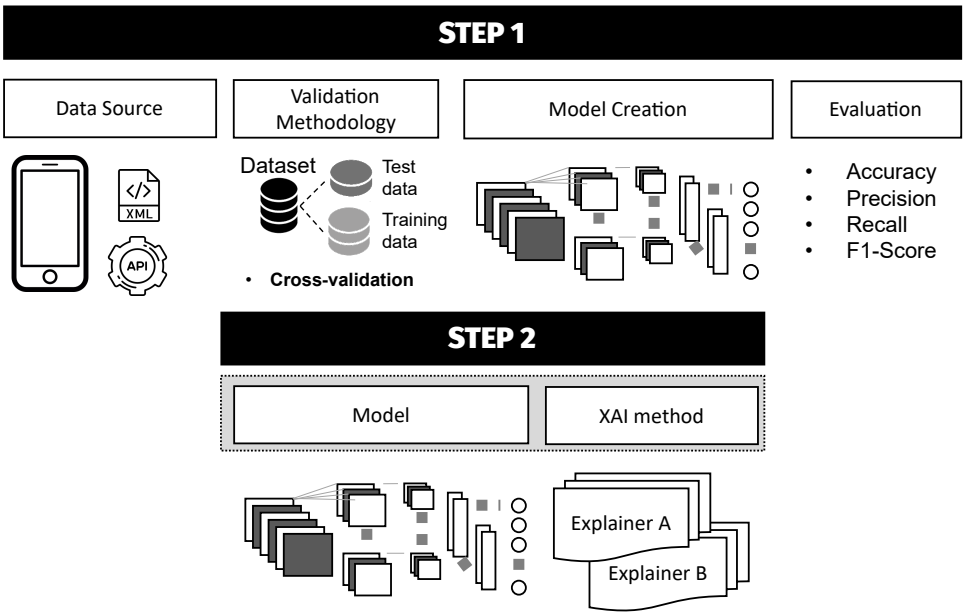

Figure 1. Overview of the XAI methodology applied to machine learning.

Any machine learning task begins with the data source. Data can come from various sources, including databases, files, APIs, online repositories, and collection tools. The data source selection is based on the specific problem we are addressing. It is imperative to ensure that the data is relevant, high-quality, and sufficient to facilitate the development of accurate and robust detection models. Preprocessing techniques may be applied to clean and transform the data into a suitable format for subsequent ML algorithms.

After obtaining and preprocessing the data, the following step is to select a validation methodology, dividing the dataset into sections for training and testing the model. The holdout method is the most basic form of validation, dividing the dataset into two sets.

If the validation sets are established, the subsequent stage involves selecting and training a machine learning algorithm using the training data. The choice of algorithm depends on various factors, including the nature of the problem, the type of data available,

and the specific requirements of the research. Once the model has been trained, assessing its performance on unseen data is crucial, evaluating the model's predictions on the test data and comparing them to the actual values. Standard evaluation metrics for classification tasks include the F1-score and accuracy, which provide measures of the model's effectiveness in correctly classifying instances.

In addition to performance evaluation, understanding and explaining the model's output is essential, particularly in tasks where errors could have significant consequences. This step helps shed light on why the model is making specific predictions. Various techniques can be employed to evaluate the model's results, such as permutation importance, partial dependence plots, SHAP, and LIME. These techniques offer insights into how the model makes decisions by facilitating the interpretation of the contribution of each feature to the model's predictions.

3.1. Why is it important to explain model results?

In the malware detection domain, the consequences of incorrect predictions can be significant, making the need for accurate and reliable models crucial. False positives occur when harmless software is wrongly labeled as malware, which can result in high costs and interruptions. False negatives, on the other hand, occur when malicious malware is wrongly labeled as benign and can result in catastrophic security breaches, data theft, and other consequential consequences.

Incorporating explainability techniques into malware detection using ML models is an important approach to get new insights about the decision-making process. Explainable AI methods enhance human understanding and trust in ML systems by providing users with tools to evaluate and comprehend model outputs.

One key benefit of XAI is the promotion of trustworthiness. This trust is critical in areas such as malware detection, where system actions directly influence users' digital safety. Understanding why a particular application was flagged as a threat can help improve malware detection algorithms, resulting in more robust and reliable security solutions. XAI also helps ML models be validated. If a model can articulate how it arrived at a particular decision, validating its efficacy and accuracy is simpler. This validation procedure is essential for maintaining the model's development and guaranteeing it functions as intended.

In the subsequent sections, we use the SHAP framework to provide explainable results and insights into the decision-making process of the malware detection system, thereby enhancing the interpretability of the system.

4. Experimental Protocol

In this section we outline the experimental protocol, which encompasses two evaluation scenarios. We provide details of the datasets utilized, the evaluation metrics employed, the baseline model, the validation methodology, and the specific evaluation scenarios.

4.1. Android Datasets

We used in our experiments seven widely-used datasets in the literature for evaluating malware detection on the Android platform. Table 1 provides information on the number of samples and the types of features present in these Android malware datasets.

Table 1. Summarization of datasets.

Dataset	Features		Samples		
	N. Features	**Feature Type**	**Malwares**	**Benign**	**Total**
AndroCrawl	81	API Calls (24) Intents (8) Permissions (49)	10170	86562	96732
ADROIT	166	Permissions	3418	8058	11476
Android Permissions	183	Permissions	20000	9999	29999
DefenseDroid	2938	Permissions (1490) Intents (1448)	6000	5975	11975
DREBIN-215	215	API Calls (73) Permissions (113) System Commands (6) Intents (23)	5555	9476	15036
KronoDroid Disp. Real	246	Permissions (146) System Calls (100)	41382	36755	78137
MH-100K	24833	API Calls (24417) Intents (250) Permissions (166)	9800	92134	101934

It is important to note that each dataset has unique features and sources, ensuring diversity in the samples and enabling a thorough evaluation of the malware detection algorithms.

4.2. Classification Model

In this work, we use the XGboost classifier (Extreme Gradient Boosting), which has been adopted in works related to malware detection [Palša et al., 2022] and intrusion detection systems [Devan and Khare, 2020]. It is a highly recommended choice for binary classification tasks involving categorical features due to its advantages over traditional machine learning models [Chen and Guestrin, 2016]. The XGBoost builds new predictors to reduce the residual errors of the prior predictor, hence increasing prediction accuracy over time. Additionally, it offers regularization to prevent overfitting, a significant problem with decision tree-based models and can handle categorical features effectively.

4.3. Evaluation Metrics and Validation Methodology

In assessing the performance of models, we employ commonly used evaluation metrics, including accuracy, precision, recall, and F1-Score. These metrics are derived from the confusion matrix analysis and provide valuable insights into the effectiveness of the classification system. While accuracy is a commonly used metric, it may not be sufficient, especially when dealing with class imbalance, as it can impact the interpretation of results.

As validation methodology, we use hold-out, known as the simplest form of splitting data and relies on a single split of the dataset into two mutually exclusive subsets called a training set and a test set. The advantage of this method is the lower computational load. We choose to use a split of 80% for training and 20% for testing.

5. Results and Discussion

In this section, we examine results from two evaluation scenarios that critically impact how we evaluate malware detection algorithms. Firstly, we assess a malware detection system's classification performance across various datasets. The XGBoost classifier is

utilized to construct models on these datasets, facilitating the differentiation between benign and malicious applications. Secondly, we employ the SHAP swarm plot visualization technique to explain how our models make decisions and highlight the importance of essential features in the prediction process.

5.1. The effectiveness of malware classification methods

We summarize the classification performance of the malware detection system on different datasets in Table 2. The results demonstrate that the malware detection models trained on the Drebin, AndroCrawl, KronoDroid, and MH-100K datasets achieved superior performance, with accuracy scores exceeding 0.97. For benign applications, the accuracy, precision, recall, and F1-score results of the Drebin, AndroCrawl, and MH-100K datasets are all close to 0.99, indicating that they provide the best results. These findings indicate that these datasets offer the best results, exhibiting minimal false positives and negatives. Thus, the models built with these datasets can differentiate between benign and malicious apps.

Table 2. Classification performance using holdout methodology for malware detection models on different datasets.

Dataset	Class	Precision	Recall	F1-score	Accuracy	Macro-F1
Drebin	Benign	0.99	0.99	0.99	0.99	0.99
	Malware	0.99	0.99	0.99		
AndroCrawl	Benign	0.99	0.99	0.99	0.99	0.97
	Malware	0.94	0.94	0.94		
KronoDroid	Benign	0.96	0.98	0.97	0.97	0.97
	Malware	0.98	0.97	0.97		
AndroidPermissions	Benign	0.55	0.13	0.21	0.67	0.50
	Malware	0.68	0.94	0.79		
Adroit	Benign	0.91	0.97	0.94	0.91	0.89
	Malware	0.92	0.78	0.85		
DefenseDroid	Benign	0.91	0.93	0.92	0.92	0.92
	Malware	0.93	0.90	0.92		
MH-100K	Benign	0.99	0.99	0.99	0.98	0.94
	Malware	0.87	0.90	0.89		

The Drebin, AndroCrawl, and MH-100K datasets yield high precision, recall, F1-score, and accuracy for both malware and benign classes, as shown in Table 2. The KronoDroid and DefenseDroid datasets produce reasonably good performance metrics in both malware and benign classes, with the models achieving balanced performance consistently, as showed by the Macro-F1 scores. The Adroit dataset also demonstrates good precision and recall for the benign class but much lower recall for the malware class.

The AndroidPermissions dataset shows inferior performance, particularly in the benign class, where it achieves an F1-score of only 0.21. Although permissions can be indicative of an app's activity, in the case of the Android Permissions dataset, they may not be sufficient for successful malware detection. Additionally, the model can become biased towards one class when the dataset has a high proportion in that class, which could lead to poor performance for the underrepresented class. Finally, smaller datasets may not include enough information for the model to generalize effectively, resulting in poor

performance. Combining permissions with other aspects like API requests and Intents is frequently a more efficient technique.

Our findings imply that the feature set and the proportion of malicious and benign samples in the datasets have a considerable impact on how well malware detection models work. Additionally, the high accuracy of the models on the MH-100K dataset suggests that a rich feature set can be crucial for reliable malware identification. The variation in performance between datasets emphasizes how crucial it is to choose the right features and have a balanced dataset for accurate malware identification.

5.2. A closer look into malware classification results

We use a visualization approach called the SHAP swarm plot to gain insights into the outputs of malware classification models. The SHAP swarm plot provides valuable information about the importance of features in determining whether a sample is classified as malware. A positive SHAP value indicates that a feature's presence or higher value contributes to the likelihood of the sample being classified as malware. Conversely, a negative SHAP value suggests that a feature's absence or lower value contributes to the classification of the sample as benign. We focused on the top 30 features that play a significant role in the decision-making process of the models. Figures 2, 3, 4, 5, 6, 7, and 8 show the SHAP values of these features on the datasets.

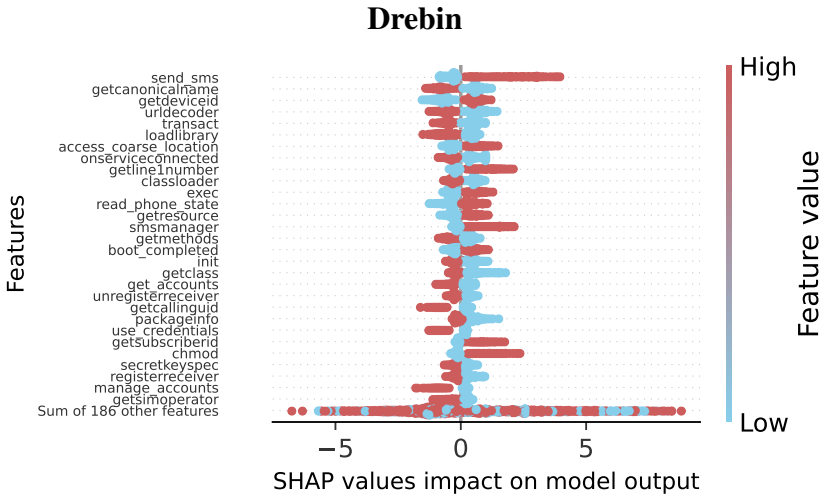

Figure 2. SHAP swarm plot for Drebin dataset.

These datasets share common feature categories, indicating similar types of system interactions. For instance, features related to accessing system-level information, such as $read\_phone\_state$), are present in Adroid, DefenseDroid, Drebin, and KronoDroid datasets. Permission-related features like $access\_coarse\_location$, $access\_network\_state$, and $access\_wifi\_state$ are also standard across some datasets, showing a shared focus on applications permissions.

The absence of a straight feature intersection demonstrates the extensive range of Android malware behaviors, demanding different and multidimensional feature sets for complete analysis and detection. However, shared feature categories indicate typical Android malware behaviors and exploitation tactics, underscoring the importance of these factors in malware identification. The SHAP framework has drawn attention to these

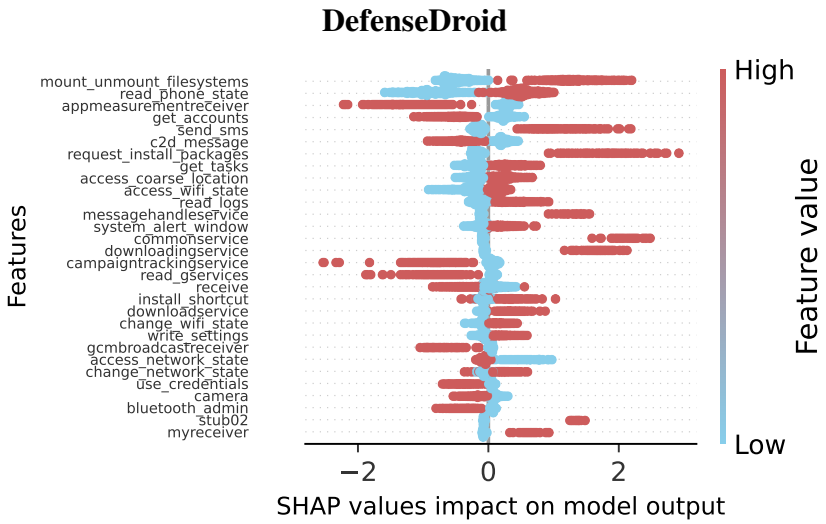

Figure 3. SHAP swarm plot for DefenseDroid dataset.

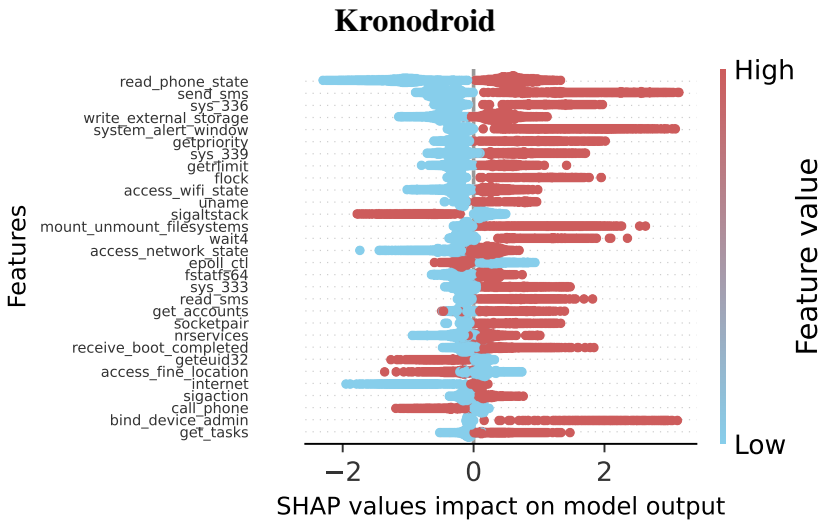

Figure 4. SHAP swarm plot for Kronodroid dataset.

crucial elements, such as system interactions, permissions, and API calls, which malicious applications often exploit or manipulate in the context of Android malware detection.

Features such as $read\_phone\_state$ and $send\_sms$, in Figures 2, 3, 4 and 6, are associated with Android system processes and states. Malicious applications usually attempt to obtain sensitive information or alter system behavior. For instance, reading the smartphone status can provide information about the device, the user, or ongoing calls, while sending an SMS can be misused for activities like signing up for premium services without the user's knowledge or sending phishing messages.

Application permissions such as $access\_coarse\_location$, $access\_network\_state$, and $access\_wifi\_state$ play a key role. While malware applications can utilize these features, their presence does not necessarily imply that the application is malicious, as they can also be used for legitimate purposes. However, their misuse or atypical usage often signifies malicious behavior, making them crucial indicators for malware detection

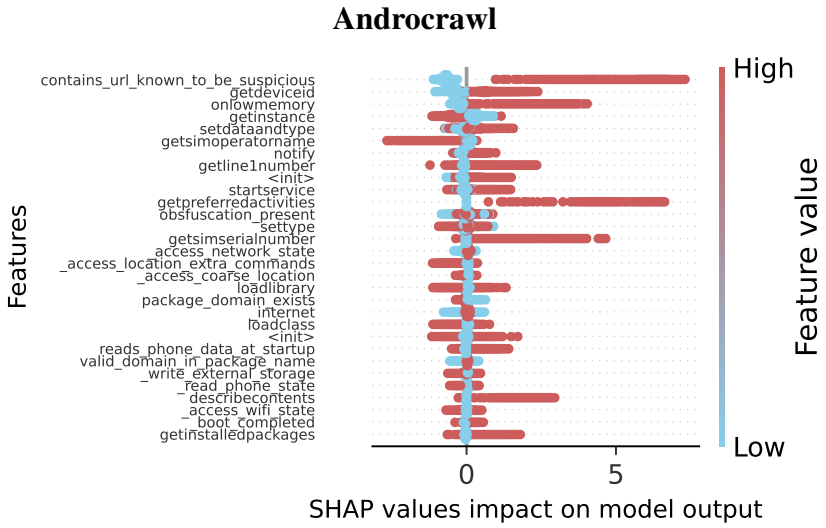

Figure 5. SHAP swarm plot for Androcrawl dataset.

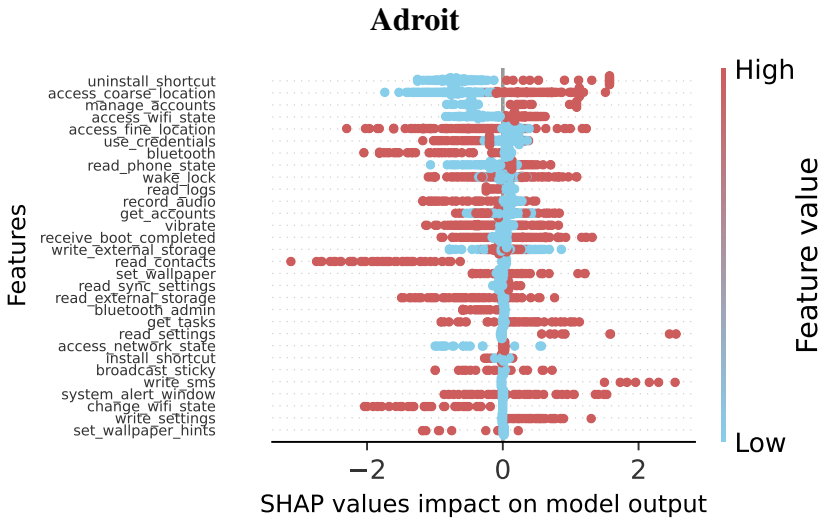

Figure 6. SHAP swarm plot for Adroit dataset.

models. Moreover, excessive or improper application permissions are warning signs of potentially harmful activity.

Malware frequently exploit specific Android API calls, as those highlighted in-Figure 8. For instance, a feature called $Landroid/os/Parcel.writeFloat()$ represents an interprocess communication API call that can be exploited to exchange private information with other processes or a command and control server. AndroCrawl dataset features like $obfuscation\_present$ are related to application properties. Malware often employs obfuscation techniques to conceal its malicious code from detection tools.

The inclusion of expert-dependent features in datasets, such as $readbrowserhistoryandbookmarks$ or $contains\_url\_known\_to\_be\_suspicious$, as shown in Figure 5 and 7, has both advantages and disadvantages. These features have the potential to improve the accuracy of detection models. This is because skilled human specialists can detect hidden red flags of malicious applications that machine learning

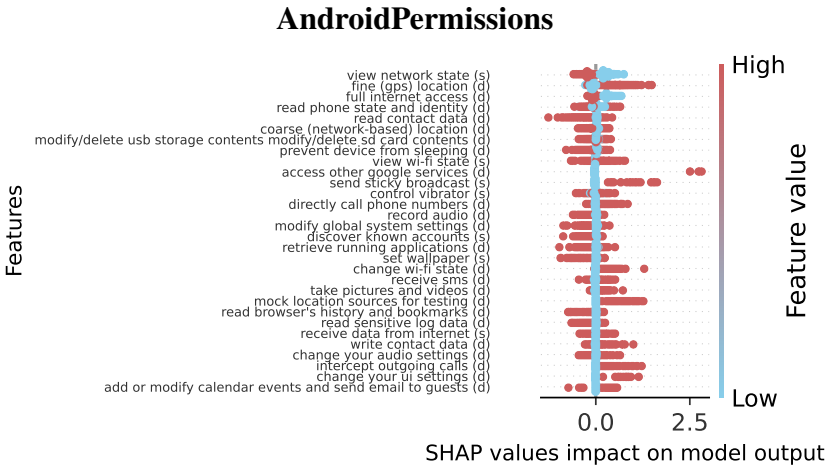

Figure 7. SHAP swarm plot for AndroidPermissions dataset.

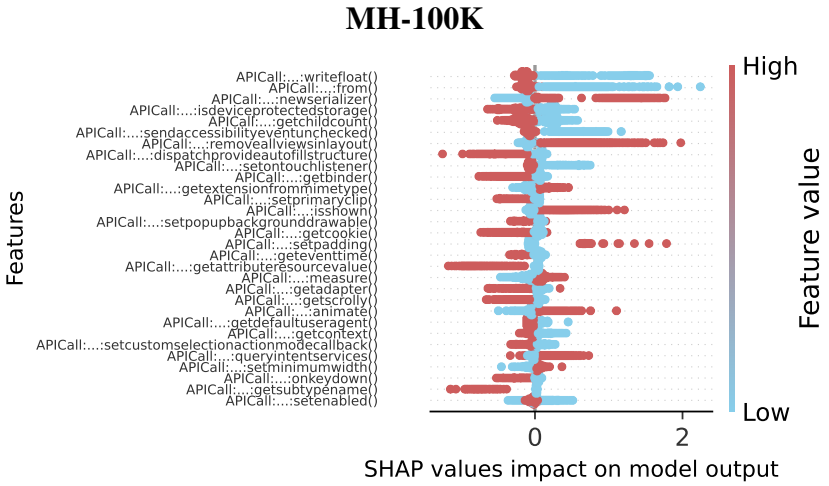

Figure 8. SHAP swarm plot for MH-100K dataset.

algorithms may miss. As a result of these expert-dependent features, malware detection algorithms can become more complex while maintaining accuracy. However, including expert-dependent features significantly increases the time and resources required for feature extraction. Unlike automatic feature extraction, which can be performed at scale and at a faster pace, expert-dependent features necessitate extensive manual review, which can be costly and time-consuming.

It is important to note that expert support can cause scaling issues. As the number of applications requiring review grows, manually extracting features from each application may become impractical or even impossible. This is especially true in the case of Android apps, where hundreds of thousands of new apps and upgrades are released each month. Another potential disadvantage is the eventual human bias. Certain features may be given disproportionate weight, while others may be missed, depending on the expert's expertise, perspective, and even personal biases. This may unintentionally introduce bias into the detection model, lowering its generalizability and impartiality.

Understanding these features and how malicious applications might use them is

essential for effective malware detection. In a machine learning model, such features are critical for determining whether an application is benign or malicious. The SHAP framework helps to highlight the importance of these features and provides valuable information for future research and mitigation efforts.

6. Conclusion

Adopting XAI approaches is a step forward in developing more transparent, trustworthy, and efficient ML models in malware detection. This evolution can improve our ability to detect and respond to evolving cybersecurity threats. While enhancing accuracy remains a critical goal, and our results demonstrate that current methods are efficient in this task, understanding the model behavior is equally important. If models make decisions based on particular features that result in unfair or biased findings, it's essential to identify these issues during the evaluation process. Such kinds of insights can lead to better model designs that provide AI systems fairness, transparency, and reliability. XAI approaches can significantly enhance this comprehension, creating a more responsible ML system.

Acknowledgement

This research was funded, as provided for in Arts. 21 and 22 of decree no. 10,521/2020, under Federal Law no. 8,387/1991, through agreement no. 003/2021, signed between ICOMP/UFAM, Flextronics da Amazônia Ltda and Motorola Mobility Comércio de Produtos Eletrônicos Ltda. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001 and by Amazonas State Research Support Foundation - FAPEAM - through the POS-GRAD project.

Data Availability Statements

The datasets used during the current study are available at GitHub (https://github.com/Malware-Hunter/SBSeg23-XAI). The MH-100K dataset is available at GitHub (https://github.com/Malware-Hunter/MH-100K-dataset) as well.

References

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T. A. E., and Elnour, A. A. H. (2022). Malware detection issues, challenges, and future directions: A survey. *Applied Sciences*, 12(17):8482.

Alani, M. M. and Awad, A. I. (2022). Paired: An explainable lightweight android malware detection system. *IEEE Access*, 10:73214–73228.

Alikhademi, G., Richardson, B., Drobina, E., and Gilbert, J. E. (2018). Can explainable ai explain unfairness? a framework for evaluating explainable ai. *arXiv preprint arXiv:1810.07339*.

Bhat, P., Behal, S., and Dutta, K. (2023). A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. *Computers & Security*, 130:103277.

Bragança, H., Rocha, V., Souto, E., Kreutz, D., and Feitosa, E. (2023). Capturing the behavior of android malware with mh-100k: A novel and multidimensional dataset. In *XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. SBC.

Charmet, F., Tanuwidjaja, H. C., Ayoubi, S., Gimenez, P.-F., Han, Y., Jmila, H., Blanc, G., Takahashi, T., and Zhang, Z. (2022). Explainable artificial intelligence for cybersecurity: a literature survey. *Annals of Telecommunications*, pages 1–24.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Colaco, C., Bagwe, M., Bose, S., and Jain, K. (2021). Defensedroid: A modern approach to android malware detection. *Strad Research*.

Devan, P. and Khare, N. (2020). An efficient xgboost–dnn-based classification model for network intrusion detection system. *Neural Computing and Applications*, 32:12499–12514.

Fan, M., Wei, W., Xie, X., Liu, Y., Guan, X., and Liu, T. (2020). Can we trust your explanations? sanity checks for interpreters in android malware analysis. *IEEE Transactions on Information Forensics and Security*, 16:838–853.

Guerra-Manzanares, A., Bahsi, H., and Nõmm, S. (2021). Kronodroid: time-based hybrid-featured dataset for effective android malware detection and characterization. *Computers & Security*, 110:102399.

Guo, W., Mu, D., Xu, J., Su, P., Wang, G., and Xing, X. (2018). Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 364–379, New York, NY, USA. Association for Computing Machinery.

Kim, H., Lee, Y., Lee, E., and Lee, T. (2021). Cost-effective valuable data detection based on the reliability of artificial intelligence. *IEEE Access*, 9:108959–108974.

Kinkead, M., Millar, S., McLaughlin, N., and O'Kane, P. (2021). Towards explainable cnns for android malware detection. *Procedia Computer Science*, 184:959–965.

Kouliaridis, V., Barmpatsalou, K., Kambourakis, G., and Chen, S. (2020). A survey on mobile malware detection techniques. *IEICE Transactions on Information and Systems*, 103(2):204–211.

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.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Martín, A., Calleja, A., Menéndez, H. D., Tapiador, J., and Camacho, D. (2016). Adroit: Android malware detection using meta-information. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Mathews, S. M. (2019). Explainable artificial intelligence applications in nlp, biomedical, and malware classification: a literature review. In *Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 2*, pages 1269–1292. Springer.

Melis, M., Scalas, M., Demontis, A., Maiorca, D., Biggio, B., Giacinto, G., and Roli, F. (2022). Do gradient-based explanations tell anything about adversarial robustness to android malware? *International journal of machine learning and cybernetics*, pages 1–16.

Mendes, C. and Rios, T. N. (2023). Explainable artificial intelligence and cybersecurity: A systematic literature review. *arXiv preprint arXiv:2303.01259*.

Miranda, T. C., Gimenez, P.-F., Lalande, J.-F., Tong, V. V. T., and Wilke, P. (2022). Debiasing android malware datasets: How can i trust your results if your dataset is biased? *IEEE Transactions on Information Forensics and Security*, 17:2182–2197.

Muzaffar, A., Hassen, H. R., Lones, M. A., and Zantout, H. (2022). An in-depth review of machine learning based android malware detection. *Computers & Security*, page 102833.

Nellaivadivelu, G., Di Troia, F., and Stamp, M. (2020). Black box analysis of android malware detectors. *Array*, 6:100022.

Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R., and Maskeliunas, R. (2018). Android malware detection: A survey. In *1st ICAI*, pages 255–266. Springer.

Palša, J., Ádám, N., Hurtuk, J., Chovancová, E., Madoš, B., Chovanec, M., and Kocan, S. (2022). Mlmd—a malware-detecting antivirus tool based on the xgboost machine learning algorithm. *Applied Sciences*, 12(13):6672.

Pimenta, T. S. R., Ceschin, F., and Gregio, A. (2023). Androidgyny: Reviewing clustering techniques for android malware family classification. *Digital Threats: Research and Practice*.

Qamar, A., Karim, A., and Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems*, 97:887–909.

Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., and Molloy, I. (2012). Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 13–22.

Scalas, M. et al. (2021). Malware analysis and detection with explainable machine learning. Technical report.

Sisto, A. (2013). AndroCrawl: Studying Alternative Android Marketplaces. Master's thesis, Politecnico di Milano.

Talbi, A., Viens, A., Leroux, L.-C., François, M., Caillol, M., and Nguyen, N. (2022). Feature importance and deep learning for android malware detection. In *ICISSP*.

Ullah, F., Alsirhani, A., Alshahrani, M. M., Alomari, A., Naeem, H., and Shah, S. A. (2022). Explainable malware detection system using transformers-based transfer learning and multi-model visual representation. *Sensors*, 22(18):6766.

Yerima, S. Y. and Sezer, S. (2019). Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE Transactions on Cybernetics*, 49(2).

Yin, M., Wortman Vaughan, J., and Wallach, H. (2019). Understanding the effect of accuracy on trust in machine learning models. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–12.

Zakeya, N., Ségla, K., Chamseddine, T., and Alvine, B. B. (2022). Probing androvul dataset for studies on android malware classification. *Journal of King Saud University-Computer and Information Sciences*, 34(9):6883–6894.