Detecting Temporal Inconsistency in Biased Datasets for Android Malware Detection

***Abstract*—Machine learning (ML) has exhibited great potential in Android malware detection. Yet, the reliability of these ML models, as well as the fairness of their evaluation, hinge significantly on the quality of the datasets used. A significant issue compromising these aspects is the presence of temporal inconsistencies within datasets, which could lead to overestimated detection performance. While previous research has acknowledged the impact of temporal inconsistencies, the proposed detection approaches often falter in accuracy and practicality. Previous studies have had limitations when it comes to dealing with complex cases of temporal inconsistencies. Additionally, their approaches require knowledge of a dataset's temporal attributes, which is often not realistic in real-world applications. In response to these challenges, we propose a novel ML-based approach to comprehensively and effectively detect temporal inconsistencies in Android malware datasets, regardless of the magnitude of these inconsistencies. Distinguishing itself from prior attempts, our approach accurately identifies inconsistencies in unknown datasets, without making any assumptions about their temporal attributes. Moreover, we introduce a new benchmark dataset of 78,000 diverse Android samples, spanning malware to benign samples from 2010 to 2022, for exploring temporal inconsistency. A rigorous evaluation of our approach using this dataset reveals its proficiency in managing temporal inconsistencies, achieving a remarkable 98.3% detection accuracy. We further validate the efficacy of our feature selection procedure and demonstrate the robustness of our approach when applied to unknown datasets. Collectively, our findings pioneer a novel performance standard in Android malware detection assessments, contributing to the enhancement of reliability in ML-based techniques.**

I. Introduction

Machine Learning (ML) techniques have been extensively adopted as effective approaches for Android malware detection [1], [2]. Trained on large and representative datasets, these ML models have shown success in distinguishing between benign and malicious apps. However, numerous studies [3], [4] have suggested that unrealistic experimental designs can introduce substantial biases, resulting in overly optimistic performance evaluations. Some claims from the literature even suggest near-perfect detection accuracy [5], [6], [2], a performance that is often an overestimation of the model's actual capabilities [3].

One of the recurring biases, temporal inconsistency, significantly impacts ML-based malware detection models [3], [4], [7]. Temporal inconsistency arises when malware and benign samples selected for training the models span different time periods. For example, Khoda *et al.*'s study [8] employed the Drebin [9], [10] and Genome datasets, which comprised malware samples collected prior to 2016, while the benign samples were gathered more recently, leading to a temporal inconsistency in the dataset. This disregard for the temporal attributes can mislead the model's learning, making it derive patterns from the samples' temporal attributes rather than their inherent malicious or benign characteristics [4]. As a result, the models' performance metrics might reflect their ability to recognize these temporal differences rather than the actual malicious or benign behaviors. Despite the evident impact of temporal inconsistency, no established method exists to detect and mitigate this issue in biased datasets, posing a significant research gap and an opportunity for further investigation.

Prior work [4], [11] has mentioned ideas about how to identify temporal inconsistency. Specifically, these studies have been limited in their ability to handle obvious cases of temporal inconsistencies, and their approaches often require prior knowledge of app samples' temporal attributes - a requirement that is often unrealistic in real-world scenarios. In this work, we develop a novel ML-based approach designed to effectively identify temporal inconsistencies in Android malware datasets, which can be applied to any unknown dataset and has better performance. This approach distinguishes itself by accurately detecting inconsistencies in unknown datasets without making unrealistic assumptions about their temporal attributes. Specifically, our innovative approach employs machine learning techniques to identify temporal features in the dataset. We then propose a set of time-sensitive features that can serve as a foundation for future investigations in this area. We further utilize a Support Vector Machine (SVM)-based classifier to examine these features and use the results to create two distinct metrics for detecting temporal inconsistency (i.e., a performance-based metric and a feature-importance-based metric). To evaluate the detection performance of our approaches, we also introduce a new benchmark dataset comprising 78,000 diverse Android samples, ranging from malware to benign applications collected from 2010 to 2022, specifically designed for the study of temporal inconsistency. Our experiments demonstrate that our approach can accurately detect temporal inconsistency in a dataset with an accuracy rate of up to 98.3%. We found that the composition and size of the feature set significantly impact temporal inconsistency detection. When applied to other datasets, our approach still maintains a high level of performance, demonstrating its robustness. In this paper, we make the following contributions:

- **Dataset**: We publish a new dataset for studying temporal inconsistency in Android malware detection, covering 78,000 malware and benign samples across different periods.
- **Approach**: We introduce a novel ML-based approach

for identifying temporal inconsistency in Android malware datasets, enhancing the accuracy and trustworthiness of ML-based malware detection. We publish the studied dataset and make it publicly available at https://anonymous.4open.science/r/MalwareDetection-B2A2/.
- **Evaluation**: We provide a comprehensive evaluation of our method using our new dataset, demonstrating its effectiveness in addressing temporal inconsistency, setting a new standard in the field.

II. Background

The growing sophistication of Android malware and the increasing complexity of Android applications underscore an urgent need for robust and reliable malware detection techniques. Machine learning-based techniques have shown great promise in this domain, achieving remarkable detection performance. However, the effectiveness of these methods hinges on the quality of the data they are trained on. Prior studies [3], [4] have proven that it's evident that certain biases and inconsistencies within the training data can materially distort the performance of the models, potentially yielding misleading or excessively optimistic outcomes.

One such bias, temporal inconsistency, has been identified as a critical factor that can distort the performance metrics of these models. Prior studies, such as those conducted by Liu *et al.* [4], have proven the extent to which temporal inconsistency between malware and benign samples can inflate detection performance. Moreover, they showed that when temporal inconsistency permeates the data, the models may inadvertently learn to differentiate samples based on their temporal features, as opposed to the traits that genuinely differentiate benign and malicious behaviors. Despite the clear evidence of the impact of temporal inconsistency, there is currently no established method to detect and address this issue in biased datasets. This represents a significant research gap in the field, which, if filled, could substantially enhance the accuracy and reliability of ML-based malware detection. To address it, we propose a novel approach to identify temporal inconsistency for the Android malware datasets and attempt to answer the following three research questions:

(RQ1) Can we replicate the impacts of temporal inconsistency? This research question seeks to demonstrate the effect of temporal inconsistency on ML-based malware detection by applying the idea proposed by Liu *et al.* [4]. We aim to examine whether their ideas could be applied to temporal inconsistency detection in an unknown dataset.

(RQ2) How effectively can the proposed metrics identify the temporal inconsistency in biased datasets? The second research question intends to evaluate the effectiveness of our proposed metrics in identifying temporal inconsistencies. As these inconsistencies frequently appear in biased datasets, a robust testing criterion that can accurately detect them is vital to ensure the integrity of machine learning-based malware detection.

(RQ3) How generalizable is our proposed approach? The motivation for this research question is to evaluate the generalizability of the proposed approach. Given the constraints in the volume and diversity of apps available for testing, it's critical to assess the robustness of the proposed approach against datasets beyond those used in its development. This evaluation will provide valuable insights into the method's potential reliability and usefulness in real-world scenarios.

III. Study Design

A. Dataset Construction

Our dataset was derived from AndroZoo [12], a large and comprehensive collection of Android applications, each with detailed metadata. We collected equal quantities of benign and malicious Android apps for each year from 2010 to 2022, amassing 3,000 instances of each annually. This balanced collection strategy, in line with previous studies [13], [14], [15], helps ensure an unbiased representation of both benign and malicious apps in our dataset. AndroZoo provides metadata for each application, including the number of positive detections (p) by anti-virus programs on VirusTotal [16]. We used this information to classify applications, with $p = 0$ indicating a benign app and $p > 4$ signifying malware, following the methodology of earlier research [3], [17]. Finally, our comprehensive dataset comprises a balanced collection of 39,000 benign and 39,000 malware applications, providing a robust base for training and evaluating ML models.

B. Data Processing

The raw APK files required transformation into a format suitable for machine learning models. This process involved extracting relevant information that could differentiate malware from benign applications. Consistent with earlier work [14], [4], we used APKtool [18] and Androguard [19] to decompile the APK files and extract static features, specifically the invoked API calls and declared permissions. These features were then converted into binary vector representations using one-hot encoding. In this schema, the presence of a specific API or permission in an application is denoted by '1' and its absence by '0'. For the sake of consistency, all vectors were made equal in length, matching the total count of unique APIs and permissions found in the entire dataset. Given the high dimensionality of the data and the necessity of computational efficiency during model training and evaluation, we limited the feature set for each classification task. We chose to incorporate only the top 2000 most frequently occurring features in the dataset. This approach allows us to focus on the most pertinent characteristics without undermining the model's ability to distinguish effectively between benign and malware applications. Our selected features can be accessed in the supplementary material online.

C. Feature Selection and Construction

1) Feature Importance Assessment: Our methodology starts with the use of interpretable machine learning models. In this approach, every feature in the model is characterized by three properties: its name, its importance, and its ranking. The feature's name corresponds to a specific API or permission

(e.g., SEND_SMS). The importance of a feature is measured by how much it contributes to the model's prediction. In our analysis, we consider the absolute value of the importance to represent the impact of the feature on the prediction. The ranking of each feature is sorted according to the absolute value of its importance.

2) Temporal Feature Extraction: The next step is to extract time-sensitive features. We use a linear machine learning model, specifically a Support Vector Machine (SVM), for this purpose, following in the footsteps of previous research [9], [3], [20]. We train the SVM classifiers on apps from different years, ensuring that they are of the same type (either benign or malicious).

In order to identify time-sensitive features, we label the apps with the year of their creation. The model's goal is to correctly identify the creation year of an app. Given that our dataset extends from 2010 to 2022, we train the model for each year difference, resulting in a total of $2 * \sum_{i=1}^{12} i = 156$ training instances. After each training session, we select the features with the highest importance, as determined by the feature weight (ω_i) in the SVM model. For each year difference, we derive two sets of important features — one from the classification of malicious apps and another from benign apps. The common set of features is obtained by intersecting these two sets. By repeating this process for each year difference, we obtain 78 groups of features ($156/2$), each group corresponding to a unique year combination. The final feature set is assembled by merging these 78 groups, producing a set of features that exhibit temporal inconsistency but do not show malicious inconsistency (a feature shows temporal inconsistency if it appears frequently in a malicious dataset but infrequently in a benign one).

The significance of this process lies in identifying the features that are crucial for classifying apps from different years and demonstrate temporal inconsistency. These features play a key role in distinguishing and understanding the variations over time in application classification. We find features with temporal inconsistency by extracting the most important features from classifications within benign or malicious apps. After intersecting the important features from both benign and malicious apps, we can identify those features that are common to both and do not exhibit a bias towards malicious behavior.

D. Metrics for Temporal Inconsistency Detection

To identify the temporal inconsistency arising from biased datasets, we propose two categories of evaluation metrics: performance-based metrics, denoted by $\mathcal{P}$, and feature importance-based metrics, denoted by $\mathcal{F}$.

Within the category $\mathcal{P}$, we have incorporated five different evaluation measures that are pivotal in assessing the performance of a model: accuracy ($\mathcal{P}_{Acc}$), F1 score ($\mathcal{P}_{F1}$), recall ($\mathcal{P}_{Rec}$), precision ($\mathcal{P}_{Pre}$), and the average of these four metrics, denoted as average performance ($\mathcal{P}_{AP}$). These diverse metrics facilitate a comprehensive evaluation of model performance from multiple dimensions, catering to various aspects of predictive ability.

Under the category $\mathcal{F}$, we have established three distinct metrics that focus on feature importance within the model and its collective influence on predictions.

1) **Average Feature Importance ($\mathcal{F}_{AI}$)**: This is the mean importance across all features. Mathematically, it is expressed as: $\mathcal{F}_{AI} = \frac{\sum_{i=1}^{n} |\beta_i|}{n}$
2) **Time-sensitive Feature Importance ($\mathcal{F}_{FI}$)**: This metric computes the total importance of features within our specifically defined time-sensitive feature set $\mathcal{T}$, as outlined in Section III-C2. It is given by: $\mathcal{F}_{FI} = \sum_{f_i \in \mathcal{T}} |\beta_i|$
3) **Outside Feature Importance ($\mathcal{F}_{OFI}$)**: To evaluate the relative impact of features not included in our time-sensitive set, we introduce this additional metric. It is computed as: $\mathcal{F}_{OFI} = \frac{1}{|\{i: f_i \notin \mathcal{T}\}|} \sum_{f_i \notin \mathcal{T}} |\beta_i|$
4) **Time-sensitive Feature Importance and Rank ($\mathcal{F}_{FIR}$)**: This metric, based on the feature set $\mathcal{T}$, further considers the values of importance and rank(mentioned in Section III-C1), using an appropriate mathematical model to represent the inconsistency of the feature. The expression is: $\mathcal{F}_{FIR} = Sig(\frac{\sum_{f_i \in \mathcal{T}} m(e/m)^{-\mathcal{R}_i} e^{|\beta_i|-1}}{\max_{f_i \in \mathcal{T}} m(e/m)^{-\mathcal{R}_i} e^{|\beta_i|-1}})$ $Sig(x) = \frac{1-e^{-x}}{1+e^{-x}}$

Within these equations, f_i denotes the i-th feature, while β_i and $\mathcal{R}_i$ symbolize the significance and ranking of the i-th feature within the model. Additionally, n and m represent the total number of features and a constant, respectively. The sets $\mathcal{T}$ and its complement consist of the features within and outside of our time-sensitive feature set, respectively. These metrics collectively facilitate a more comprehensive analysis of feature importance in our model. The output values of these two types of metrics will be between 0 and 1. To have a unified standard for predicting bias in unknown datasets, we need to define a threshold Θ for each metric. When the output is greater than Θ, it's considered that the dataset has temporal inconsistency; otherwise, it's considered that the dataset is normal.

IV. Result

A. RQ1

We aim to build upon the research conducted by Liu *et al.* [4], by replicating their methodology and, in doing so, affirming the impacts of temporal inconsistency in ML-based malware detection. Additionally, our goal is to scrutinize the effectiveness of Liu *et al.*'s approach in accurately identifying temporal inconsistencies.

Experiment setup. Adhering to the methodology of Liu *et al.* [4], we replicated their experimental steps. Two different scenarios were tested: the first combined 2010 benign dataset with each malware dataset ranging from 2010 to 2020, and the second combined 2010 malware dataset with each benign dataset from 2010 to 2020. The experiment began with training and testing our dataset, from which we were able to glean the accuracy and F1 score. Then we identify the top 10

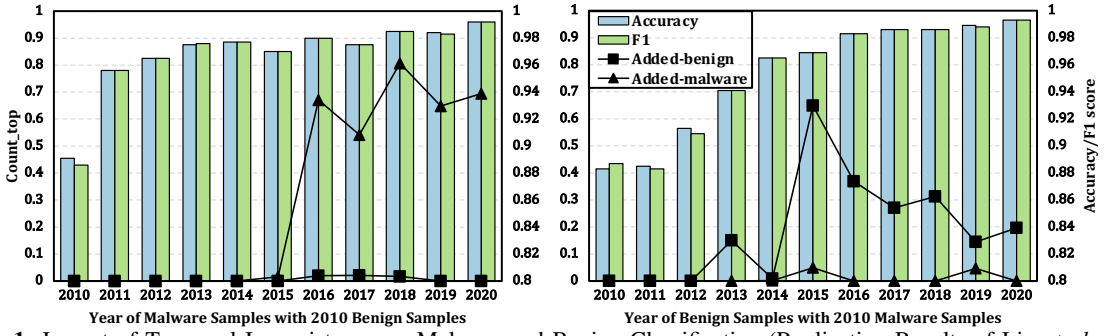

Fig. 1: Impact of Temporal Inconsistency on Malware and Benign Classification (Replication Results of Liu *et al.* [4])

most important features. We then delved into the Android development documentation to identify features that were only introduced in later versions due to Android API updates, hence leading to temporal discrepancies between datasets of malware and benign apps. These Liu *et al.*term "added features". The final step involved calculating the proportion of samples containing these added features within the respective malware and benign apps datasets.

Results. Figure 1 shows the replication results. The left side presents the classification results of benign apps from 2010 contrasted with malware from 2010 to 2020. Conversely, the right side shows the results of malware from 2010 compared with benign apps from 2010 to 2020. It is evident that the performance of the classifier improved as the temporal inconsistency increased (i.e., accuracy increases from about 0.45 to 0.95). This finding confirms the impact of temporal inconsistency in malware detection, in line with the observations of Liu et al. [4]. Moreover, our results indicated that the disparity in the proportion of samples containing added features between benign apps and malware could reflect the presence of temporal inconsistency.

However, we must acknowledge certain limitations of this approach. When the temporal bias is relatively minimal (e.g., less than 5 years), the 'added features' method struggles to provide a clear indication of the extent of temporal inconsistency. Furthermore, this approach assumes that the year of the dataset is known, enabling the identification of specific added features. However, in real-world scenarios, the specific year of a dataset under testing might be unknown, potentially reducing the effectiveness of this method. At the same time, this approach often can not identify temporal inconsistency even if large time span exists. Thus, our findings underscore the need for developing alternative methodologies to accurately detect temporal inconsistencies in datasets of indeterminate chronological origin.

B. RQ2

The main goal of this study is to thoroughly evaluate the efficacy of the proposed evaluation metrics in accurately identifying temporal bias within the datasets.

Experimental Setup. We have meticulously constructed an experiment to effectively illustrate the proficiency of our proposed method in identifying temporal bias across a variety of datasets. We employ Support Vector Machine (SVM) as our classification algorithm, given its robustness and broad applicability [3], [4].

The dataset we utilized in this study contains Android applications spanning from the years 2010 to 2022. We performed permutations and combinations between malicious and benign applications, labeling the resultant datasets with the magnitude of their temporal bias. For example, a dataset combining malicious applications from 2012 with benign applications from 2015 will have a temporal bias of 3 ($2015 - 2012 = 3$). This approach allowed us to generate a multitude of datasets with their respective temporal biases. Datasets with a temporal bias of 0 were classified as unbiased, while all others were considered as biased.

A Comparison Analysis of Different Metrics. For each metric, we generated corresponding box plots (as illustrated in Figure 2). On these plots, the horizontal axis signifies the inherent temporal bias of the dataset, while the vertical axis corresponds to the values yielded by each respective metric.

Upon comparing the two metric categories, we observed that feature importance-based metrics ($\mathcal{F}$) displayed a considerable overlap between the box with zero temporal bias and the other boxes. For instance, the performance-based metric $\mathcal{P}_{ACC}$ demonstrated a significant overlap in box plots across temporal biases of 0, 1, and 2. This overlap, essentially indicating similar metric values for distinct temporal bias levels, poses a considerable challenge in distinguishing datasets with zero temporal bias from those with slight temporal biases. In comparing the performance-based metrics ($\mathcal{P}$) with feature importance-based metrics ($\mathcal{F}$), we observed a substantial overlap in the box plots representing zero temporal bias and non-zero biases for the feature importance-based metrics. For example, the box plots for $\mathcal{F}_{FI}$ and particularly $\mathcal{F}_{FIR}$, effectively confined the box representing zero temporal bias to a very narrow range. This helped in clearly distinguishing it from the boxes representing non-zero biases. This divergence underscores the importance of time-sensitive feature sets in accurately identifying temporal inconsistencies in datasets. For each metric, we define the upper quartile of outputs from temporal bias = 1 as Θ.

Table I presents the performance of the investigated metrics in detecting temporal inconsistencies. Remarkably, $\mathcal{F}_{FIR}$, employing a more sophisticated mathematical model, achieves the highest performance. Its superior results suggest its mathematical model's increased efficiency in capturing temporal incon-

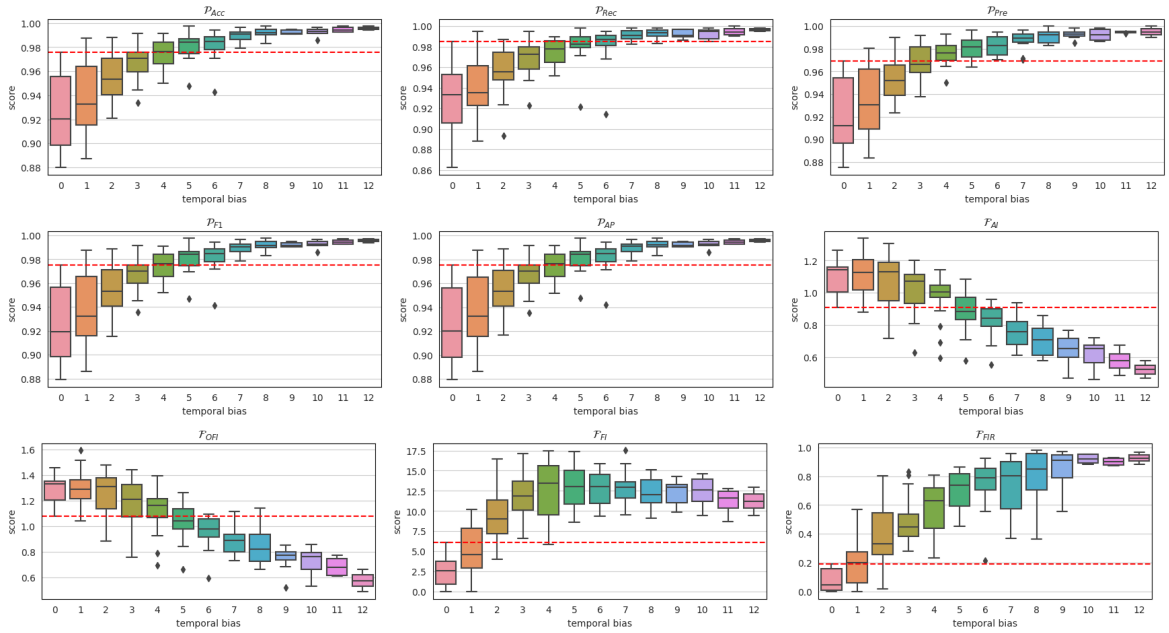

Fig. 2: Box Plots of the Accuracy of Temporal Inconsistency Detection

Metric	Θ	Accuracy	Recall	Precision	F1-score
$\mathcal{P}_{Acc}$	0.96	69.8%	84.6%	18.3%	30.1%
$\mathcal{P}_{Pre}$	0.96	74.0%	92.3%	21.8%	35.3%
$\mathcal{P}_{Rec}$	0.96	72.8%	76.9%	18.9%	30.3%
$\mathcal{P}_{F1}$	0.96	68.6%	84.6%	17.7%	29.3%
$\mathcal{P}_{AP}$	0.96	69.2%	84.6%	18.0%	29.7%
$\mathcal{F}_{AI}$	0.97	59.2%	76.9%	10.6%	18.7%
$\mathcal{F}_{FI}$	8.00	81.6%	100.0%	29.5%	45.6%
$\mathcal{F}_{OFI}$	1.18	65.1%	76.9%	9.62%	17.1%
$\mathcal{F}_{FIR}$	0.27	**83.2%**	**100.0%**	**31.7%**	**48.1%**

TABLE I: Performance comparison of the two metric categories for predicting the temporal bias in the dataset. Each metric has an associated threshold (Θ).

Metric	Θ	Accuracy	Recall	Precision	F1-score
$\mathcal{P}_{Acc}$	0.96	86.2%	84.6%	42.3%	56.4%
$\mathcal{P}_{Pre}$	0.96	91.9%	92.3%	57.1%	70.1%
$\mathcal{P}_{Rec}$	0.96	88.6%	76.9%	47.6%	58.8%
$\mathcal{P}_{F1}$	0.96	85.4%	84.6%	40.7%	55.0%
$\mathcal{P}_{AP}$	0.96	85.4%	84.6%	40.7%	55.0%
$\mathcal{F}_{AI}$	0.97	72.3%	76.9%	12.2%	21.1%
$\mathcal{F}_{FI}$	8.00	97.5%	100.0%	81.3%	89.7%
$\mathcal{F}_{OFI}$	1.18	80.1%	76.9%	10.9%	19.0%
$\mathcal{F}_{FIR}$	0.27	**98.3%**	**100.0%**	86.7%	**92.9%**

TABLE II: Performance comparison of metrics for detecting temporal bias, considering only datasets with a temporal bias greater than 2 as abnormal.

sistencies from feature importances. However, a noteworthy observation is the relatively low precision across all methods, which can be attributed to the low proportion of datasets with a temporal bias of 0 in our dataset. To conclude, our analysis underscores the effectiveness of feature importance-based metrics, particularly $\mathcal{F}_{FI}$ and $\mathcal{F}_{FIR}$, in identifying temporal bias.

Enhanced Evaluation. We discerned in the $\mathcal{F}_{FIR}$ plot of Figure 2 that the lower limit of the box corresponding to a temporal bias of 0 or 1 is considerably low, which contributes to the observed low precision. This likely stems from the minimal gap in years between datasets when the temporal bias is 1 or 2, resulting in fewer or no features with temporal inconsistency. To substantiate this hypothesis, we identified the most influential features from some datasets where the temporal bias equals 1 or 2, and checked their temporal inconsistency in the Android developer documentation. Our findings revealed that these datasets infrequently depend on temporal inconsistency for classification, relying instead on the inherent properties of the features. Nevertheless, for datasets with a substantial temporal bias (greater than 3), temporal inconsistency is a significant factor in classification.

To portray our method's performance more accurately, we repeated the experiment excluding datasets with a temporal bias of 1 or 2, as accurately determining the impact of temporal inconsistency on such datasets is challenging. In this refined experiment, we treated datasets with a temporal bias exceeding 2 as those manifesting temporal inconsistency, and only those with a temporal bias of 0 were deemed as normal datasets. As evidenced in Table II, our proposed metric $\mathcal{F}_{FIR}$ displays significant superiority in performance over the other metrics.

C. RQ3

Given that the construction of the feature set and the evaluation of method performance both depend on datasets, there is a potential concern regarding the generalizability of our method to new datasets when using the same dataset for both purposes. To mitigate this concern, we will conduct rigorous testing of our method using previously released datasets, specifically MalScan [21] and Drebin [9]. This approach ensures a thorough assessment of the effectiveness and robustness of our method across diverse datasets.

Experimental Setup . For validation purposes, we gathered the MalScan and Drebin datasets. MalScan comprises both benign and malicious apps (about 1800 malicious samples

Dataset	Output	Dataset	Output
MalScan2011+Drebin	0.045	AndroZoo2011+Drebin	0.041
MalScan2012+Drebin	0.107	AndroZoo2012+Drebin	0.059
MalScan2013+Drebin	0.177	AndroZoo2013+Drebin	0.173
MalScan2014+Drebin	0.332	AndroZoo2014+Drebin	0.522
MalScan2018+Drebin	0.731	AndroZoo2018+Drebin	0.576

TABLE III: The combination of datasets from different years and Drebin

and benign samples each year) dating from 2011 to 2018. From Drebin, we exclusively assembled the dataset (about 5400 samples) of malicious apps from 2010 to 2012. In our experiment, we mixed the datasets within MalScan, mirroring the approach detailed in Section IV-B, and employed our method to predict the temporal inconsistencies across the datasets. We further mixed those malware apps from Drebin with datasets (both MalScan's dataset and our dataset) from varying years to verify if the prediction outcomes are in line with the actual facts.

Results. Additionally, as outlined in Section IV-B, we established a threshold using our dataset. This threshold will be consistently utilized not only in this experiment but also in subsequent ones. To further showcase the efficacy of our method on novel datasets, we integrated the Drebin malware dataset from the years 2010 to 2012 with other datasets to validate if the output values meet our expectations.

Table III shows the result get from combinations of several datasets and Drebin. The left side represents the combination of the MalScan dataset and Drebin, while the right side represents the combination of our dataset and Drebin. The *Output* represents the output obtained from these datasets using our method. Similarly, we use the threshold of 0.27 from Section IV-B. Since the Drebin dataset is from the years 2010 to 2012, whether we use the MalScan dataset or our own dataset, the output is very small when the year falls within this range, indicating that the dataset combined in this way has almost no temporal inconsistency. As the year of the dataset gradually increases, the output also gradually increases.

V. Conclusion

In this paper, we have presented a novel ML-based approach for detecting temporal inconsistency in Android malware datasets. Our findings underscore the significance of addressing this overlooked issue to enhance the accuracy and reliability of ML-based Android malware detection. Our proposed method, coupled with the new dataset we introduced, sets a new benchmark in this field. Our study represents a significant step towards more reliable and accurate ML-based Android malware detection. By identifying and addressing the issue of temporal inconsistency in datasets, we pave the way for further advancements in this field. Future work could focus on refining our method further, exploring other potential biases in Android malware datasets, and developing more sophisticated mechanisms for bias detection and mitigation.

References

[1] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–41, 2017.

[2] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Deep learning for android malware defenses: A systematic literature review," *ACM Comput. Surv.*, vol. 55, no. 8, dec 2022. [Online]. Available: https://doi.org/10.1145/3544968

[3] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, L. Cavallaro *et al.*, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *Proceedings of the 28th USENIX Security Symposium*. USENIX Association, 2019, pp. 729–746.

[4] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Explainable ai for android malware detection: Towards understanding why the models perform so well?" in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022, pp. 169–180.

[5] X. Su, W. Shi, X. Qu, Y. Zheng, and X. Liu, "Droiddeep: using deep belief network to characterize and detect android malware," *Soft Computing*, vol. 24, no. 8, pp. 6017–6030, 2020.

[6] X. Su, D. Zhang, W. Li, and K. Zhao, "A deep learning approach to android malware feature learning and detection," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 244–251.

[7] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "{CADE}: Detecting and explaining concept drift samples for security applications," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[8] M. E. Khoda, J. Kamruzzaman, I. Gondal, T. Imam, and A. Rahman, "Mobile malware detection: An analysis of deep learning model," in *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019, pp. 1161–1166.

[9] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[10] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 95–109.

[11] Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy, "On the impact of sample duplication in machine-learning-based android malware detection," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 3, pp. 1–38, 2021.

[12] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proceedings of the 13th International Conference on Mining Software Repositories*, ser. MSR '16. New York, NY, USA: ACM, 2016, pp. 468–471. [Online]. Available: http://doi.acm.org/10.1145/2901739.2903508

[13] H. Alshahrani, H. Mansourt, S. Thorn, A. Alshehri, A. Alzahrani, and H. Fu, "Ddefender: Android application threat detection using static and dynamic analysis," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2018, pp. 1–6.

[14] A. Bacci, A. Bartoli, F. Martinelli, E. Medvet, and F. Mercaldo, "Detection of obfuscation techniques in android applications," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–9.

[15] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android iot devices using deep learning," *Ad Hoc Networks*, vol. 101, p. 102098, 2020.

[16] V. VirusTotal, "Virustotal: Free online virus, malware and url scanner," 2014.

[17] B. Miller, A. Kantchelian, M. C. Tschantz, S. Afroz, R. Bachwani, R. Faizullabhoy, L. Huang, V. Shankar, T. Wu, G. Yiu *et al.*, "Reviewer integration and performance measurement for malware detection," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 122–141.

[18] R. Winsniewski, "Android–apktool: A tool for reverse engineering android apk files," *Retrieved February*, vol. 10, p. 2020, 2012.

[19] A. Desnos and G. Gueguen, "Androguard-reverse engineering, malware and goodware analysis of android applications," *URL code. google. com/p/androguard*, vol. 153, 2013.

[20] N. Daoudi, K. Allix, T. F. Bissyandé, and J. Klein, "Lessons learnt on reproducibility in machine learning based android malware detection," *Empirical Software Engineering*, vol. 26, no. 4, p. 74, 2021.

[21] Y. Wu, X. Li, D. Zou, W. Yang, X. Zhang, and H. Jin, "Malscan: Fast market-wide mobile malware scanning by social-network centrality analysis," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 139–150.