Evaluation and Comparison of GBDT ML Models in Behavior-Based Malware Detection

Tustin Annika Choa, Julianne Amor de Veyra, Jose Miguel Escalona, Patrick Ryan Fortiz, and Jocelynn Cu

Center for Network and Information Security, College of Computer Studies,
De La Salle University – Manila, Philippines
{tustin_choa, julianne_deveyra, jose_miguel_escalona,
patrick_ryan_fortiz, jocelynn.cu}@dlsu.edu.ph

Abstract. This study evaluates the application of Gradient Boosted Decision Tree (GBDT) models—LightGBM and CatBoost—in behavior-based malware detection, addressing challenges such as limited publicly available datasets and inconsistent evaluation metrics. The research involved comprehensive dataset analysis, model development, and performance assessment, focusing on distinguishing between benign and malicious samples using API Call sequences. Results indicate that both GBDT models performed effectively, with LightGBM demonstrating a slight advantage in processing efficiency. However, the analysis revealed that API Calls alone may be insufficient in rare cases, necessitating the inclusion of DLL sequences for more accurate detection. The study underscores the importance of a balanced and high-quality dataset for reliable behavior-based malware detection, suggesting improvements in the verification process for collecting both benign and malicious samples. The findings contribute to enhancing behavior-based detection methods and establish a foundation for future research in this field.

Keywords: behavior-based malware detection, malware detection, API Calls, GBDT, Gradient Boosted Decision Trees, LightGBM, CatBoost, machine learning

1 INTRODUCTION

The rapid advancement of digital technology, particularly in personal computing, has led to a corresponding increase in cyberattacks and cybercrime. Traditionally, malware detection has relied heavily on signature-based methods, which, while effective against known threats, often fall short in identifying new and evolving forms of malware. The escalating complexity of malicious software necessitates more sophisticated detection techniques, prompting the exploration of behavior-based methods as a supplementary approach.

Behavior-based malware detection, often implemented through machine learning (ML) or deep learning (DL), offers a potential solution by focusing on the actions and behaviors of programs rather than their code signatures. However, research in this

J. Caro et al. (eds.), Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2024) , Atlantis Highlights in Computer Sciences 23,
https://doi.org/10.2991/978-94-6463-684-0_9

area faces several challenges, including the scarcity of publicly available datasets, inconsistencies in the metrics used across studies, and the limited comparison of the latest Gradient Boosted Decision Tree (GBDT) algorithms.

This study aims to evaluate and compare different GBDT implementations for behavior-based malware detection. Specifically, it seeks to address the current gaps in the literature by obtaining a suitable dataset, implementing select behavior-based models, and rigorously assessing their performance across a range of metrics. The ultimate goal is to enhance the understanding of behavior-based detection methods and their applicability in real-world scenarios.

2 Related Studies

The landscape of malware detection has evolved significantly, with numerous studies exploring both signature-based and behavior-based approaches. Signature-based malware detection, exemplified by datasets like EMBER [1] and SOREL-20M [2], has been widely adopted due to its effectiveness in detecting known threats. These datasets, containing millions of samples, serve as benchmarks in the field and have been extensively documented to validate their robustness and applicability.

In contrast, behavior-based malware detection, which focuses on analyzing the actions and interactions of software, is gaining traction as a complementary method to signature-based approaches. Publicly available datasets for behavior-based detection, such as "Malware Analysis Datasets: API Call Sequences" [3, 4] and MalbehavD-V1 [5], are more limited in scope and scale. These datasets primarily rely on features like API calls [2, 4–6], DLL calls [7], and network operations [7] but lack the extensive benchmarking seen in signature-based datasets. Consequently, there is a recognized need for further analysis to identify the strengths and weaknesses of these datasets and to establish comprehensive benchmarks.

Several studies have implemented behavior-based malware detection using a variety of datasets and machine learning models [4, 7–10]. However, inconsistencies in the datasets and metrics used across these studies complicate direct comparisons of their results. Unlike signature-based detection, where the use of standard datasets like EMBER allows for straightforward performance comparisons [1, 2, 11, 12], behavior-based detection lacks such uniformity.

Most existing research on behavior-based detection has focused on traditional machine learning models [8, 9, 13] or deep learning approaches [4, 8, 9, 13]. However, the use of advanced ensemble models, such as Gradient Boosted Decision Trees (GBDT), remains relatively underexplored. While some studies have compared different GBDT implementations, including XGBoost, LightGBM, and CatBoost [7, 13–16], none have comprehensively evaluated LightGBM and CatBoost using publicly available datasets.

This study seeks to address these gaps by evaluating and comparing the performance of LightGBM and CatBoost models in behavior-based malware detection. By focusing on a publicly available dataset and employing rigorous testing across various

metrics, this research aims to contribute to the development of standardized benchmarks and methodologies in this emerging area.

3 Research Methodologies

3.1 Time- and Instance-based Behaviors

API Calls serve as the primary behavioral data for applications, revealing patterns based on their instance, frequency, and order—collectively referred to as time-based behaviors. Time-based behaviors, though rich in context, result in heavier datasets that may be challenging for certain models. In contrast, instance-based behaviors focus solely on the instance and order of API Calls, offering a lighter dataset at the cost of contextual depth. This research explores both behavior types to assess their relative advantages and limitations, aiming to provide a comprehensive understanding of how API Calls can be utilized as features in behavior-based malware detection.

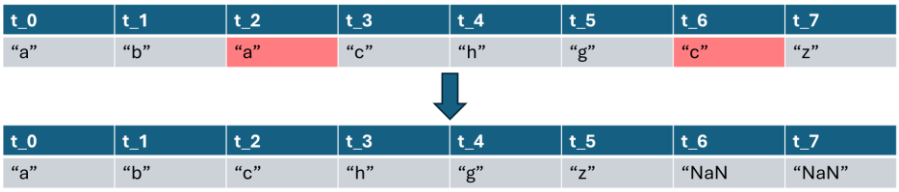

Fig. 1. A visualization of the changes made from turning API Calls on Time-based behaviors to In-stance-based behaviors where any the same succeeding API Calls are omitted to a “NaN” value.

3.2 Dataset Selection

The selection process identified a few publicly available datasets suitable for behavior-based malware detection, each designed and validated in prior studies. For this research, a single dataset was chosen based on the following criteria:

1. Inclusion of relatively recent malware samples.
2. Availability of both malicious and benign samples.
3. Use of Win32 API Calls as primary features.
4. Adequate and balanced sample sizes.
5. Suitability for exploring both time- and instance-based behavior types.

Among the considered datasets, the "Malware Analysis Datasets: API Call Sequences" [3, 4] dataset was selected for its alignment with these criteria.

3.3 Dataset Analysis

The selected dataset underwent a detailed analysis involving the identification and verification of malware types using VirusTotal. This step not only provided insights into the dataset's composition but also helped identify any falsely labeled samples, which were further examined to determine their alignment with either benign or malicious categories. The analysis extended to a comparison between API Call Patterns (time-based behavior) and API Calls (instance-based behavior), utilizing Levenshtein Distance and exploratory data analysis (EDA) techniques to uncover notable similarities and differences between malicious and benign samples.

To enhance the dataset's interpretability, Principal Component Analysis (PCA) was applied, followed by clustering via K-Means. Clusters were then visualized, and a representative 10% sample from each of the top API Call Patterns within the clusters was selected for further malware family identification using VirusTotal.

3.4 Dataset Preprocessing

The dataset used in this study comprises 42,797 malicious samples and 1,079 benign samples, with the malicious samples sourced from VirusShare and the benign samples from Windows 7 executables and PortableApps.com [3, 4]. The dataset features 100 non-consecutive repeating API Calls, supporting both time- and instance-based behaviors.

To facilitate model training and evaluation, the dataset was split into training, validation, and evaluation sets in two tranches. The first tranche divided the dataset into a 90% training/validation split and a 10% test/holdout split, with the latter reserved for assessing real-world performance. The second tranche further split the training/validation data into training and validation subsets using both static and dynamic methods, including Stratified K-Folds, to support tuning and evaluation. Additionally, to address class imbalance between malicious and benign samples, the Synthetic Minority Over-sampling Technique for Nominal attributes (SMOTEN) was employed.

3.5 Model Training and Tuning

Model training involved both default and tuned configurations for each behavior type. The models were initially trained using the statically split training data from the second tranche. Initial performance was assessed via the validation split, with behavior analysis performed using line graphs. The best-performing models were then subject to further testing during evaluation.

Hyperparameter tuning was conducted using the training and validation splits, focusing on key parameters such as the number of estimators, maximum tree depth, and regularization terms. This process employed a Stratified K-Folds technique to ensure the robustness of the tuning process across various metrics. The selection of the final tuning configuration was based on a manual evaluation of the average ranks, with eight models built in total—covering both GBDT models across both behavior types and tuning configurations.

3.6 Model Evaluation

The evaluation phase involved a comprehensive series of seven tests, each designed to assess specific aspects of model performance.

Table 1. List of Tests for Evaluating and Validating Model Performance

Test	Description
Model Robustness Test	Determines the “real-world” performance of the model using the unseen portion of the dataset used during model development.
Repeated Holdout Test	Determines if the trained models are overfitted using Stratified K-Folds method
McNemar Test	Determines if any two models (e.g., Untuned vs Tuned, LightGBM vs CatBoost) is significantly different.
Model Generalization Test	Determines model performance by training and testing it on a “unbiased” or balanced dataset and is compared alongside other non-GBDT models (SVM and MLP).
Exclusivity Test	Determines the model is reliant on the existence of API Calls identified to be unique amongst benign or amongst malicious samples.
Trojan-Undersampling Test	Determines if the number Trojan samples used during training adversely affects model performance, especially in terms of benign performance.
Benign Performance Test	Determines under what clusters or malware types is confused with when detecting benign samples.
Model Speed Test	Determines the speed of the models in fitting/training and prediction as measured in seconds and nanoseconds in a ten (10) round benchmark test.

4 Results and Analysis

4.1 Dataset Analysis

The dataset analysis reveals that it comprises eleven distinct malware types, with Trojans, Downloaders, PUAs (Potentially Unwanted Applications), and Adware (collectively referred to as "TPAD") being the most prevalent.

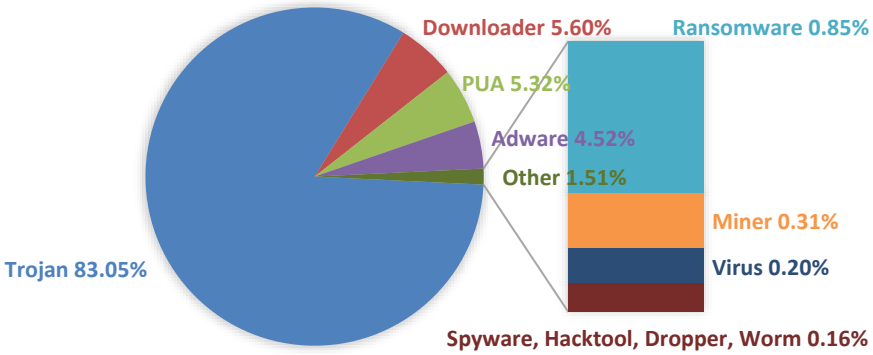

Fig. 2. The distribution of malware types of the verified malware samples of the dataset with Trojan, Downloader, PUA, and Adware being the leading malware types with at least 1,000 samples.

Upon further inspection, 6.1640% of the samples labeled as malicious, totaling 2,638 entries, were found to be falsely labeled. These mislabeled samples were excluded from the dataset, reducing the number of usable malicious samples to 40,159. The presence of these falsely labeled samples, which exhibited rare instances of overlapping API Call Patterns between benign and malicious samples, suggests potential behavioral similarities between the two categories.

Further analysis using Levenshtein Distance on API Call Patterns reinforces this observation, indicating that while most malicious and benign samples share some similarities (with similarity ratios ranging from 0.3 to 0.6), they are not identical. This finding underscores the ability of machine learning and deep learning models to discern subtle differences between benign and malicious samples. However, in rare cases where API Call Patterns are identical, API Calls, while an effective behavioral feature, may not be sufficient on their own for accurate detection.

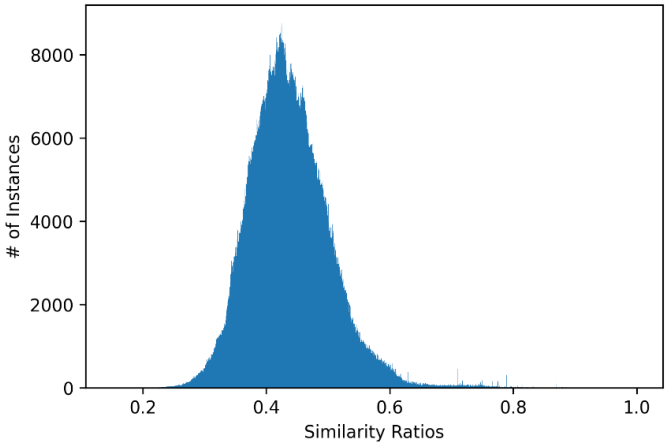

Fig. 3. The distribution of Levenshtein Distances comparing Benign and Malicious samples where most are in between 0.3 to 0.6 (i.e., similar but not the same).

Moreover, the analysis of API Calls demonstrates that malicious samples utilize a broader range of API Calls (248, with 60 being "exclusive") compared to benign samples (200, with 12 being "exclusive"). Both categories share 188 unique API Calls. It is important to note that the presence of specific API Calls does not inherently indicate malicious or benign intent, as these calls are non-partisan by nature. Therefore, additional factors, such as the order and frequency of these calls, may be necessary to accurately infer intent.

Finally, the PCA K-Means clustering results show that clustering malware samples by family is more viable, effective, and informative than by type.

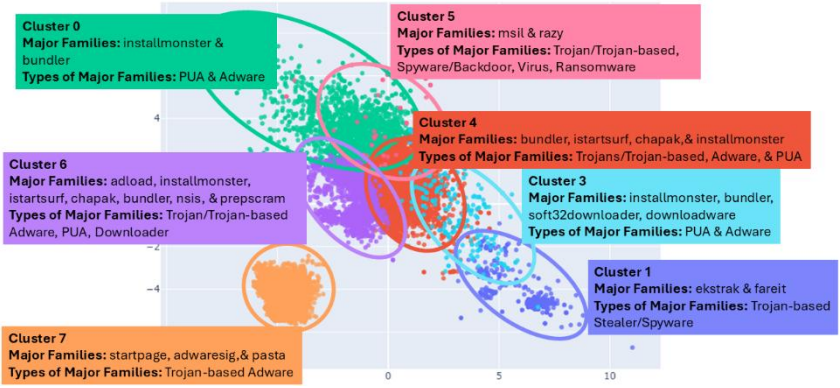

Fig. 4. The malware families identified and annotated on each cluster produced by PCA K-Means mapped by two consecutive PCA components values in x-axis & y-axis.

4.2 Dataset Preprocessing

The initial dataset split resulted in 90% of the samples being allocated for training and validation (Train Split), while the remaining 10% were reserved for testing purposes (Test/Holdout Split).

Table 2. Sample distribution between Train and Test/Holdout Split from the 1st Tranche Dataset Splitting.

	Total Quantity	Malicious Samples	Benign Samples
Train Split (90%)	37,112	36,148	966
Test/Holdout Split (10%)	4,124	4,011	113
Total Quantity	41,236	40,159	1,079

The application of SMOTEN to address the class imbalance increased the number of benign samples from 966 to 36,148. However, upon closer inspection, this synthesis turned out to be a simple replication of the minority class samples, likely due to the categorical nature of the dataset. The subsequent dataset splits using both static and dynamic methods yielded the expected distributions, with no overlapping samples observed during tuning and evaluation.

Table 3. Sample distribution between Training and Validation Split from the 2nd Tranche Dataset Splitting.

2nd Tranche Dataset Splitting	Total Quantity	Training Split	Validation Split
Static (train_test_split)	72,296	50,607	21,689
Dynamic (Stratified K-Folds)	72,296	~57,837	~14,459

4.3 Model Training and Tuning

Both Gradient Boosted Decision Tree (GBDT) models—LightGBM and CatBoost—demonstrated stable behavior across most of the training process, regardless of tuning. A minor degree of fluctuation was observed, particularly during the initial 20 iterations of model training. LightGBM consistently exhibited a faster convergence rate than CatBoost, requiring fewer iterations to achieve similar results, indicating greater efficiency in handling the given data.

Hyperparameter tuning identified several key factors influencing model performance, notably the number of estimators/trees, number of leaves, and tree depth for both models. Although the introduction of regularization slightly reduced raw performance, it was essential for achieving more stable and predictable model behavior. Specifically, Time-based LightGBM employed fewer trees with smaller leaves compared to Instance-based LightGBM, reflecting its need to compensate for the lack of information inherent to Time-based behaviors. Conversely, Time-based CatBoost required five times as many trees as its Instance-based counterpart, likely to manage the additional complexity of Time-based data.

The model training process produced eight model files, corresponding to the different combinations of behavior types, models, and tuning configurations. Analysis of these files revealed that LightGBM is consistently lighter in terms of file size compared to CatBoost across all configurations. This difference is partially attributed to LightGBM's use of encoded (integer-format) features, in contrast to CatBoost's reliance on unencoded (string-format) features, further underscoring LightGBM's efficiency.

Table 4. File Sizes of each of the GBDT Models

	LightGBM		CatBoost	
Behavior Type	Default	Tuned	Default	Tuned
Time-based Behavior	291.4102	833.3789	1701.1025	15877.9912
Instance-based Behavior	286.4102	1813.6914	1154.8369	3135.4209

4.4 Model Evaluation

The model evaluation phase involved seven tests to assess various aspects of model performance.

Model Robustness Test: Both models surpassed the study's minimum performance threshold of 80%. However, a deeper analysis revealed performance degradation when handling benign samples, primarily due to the limited quantity and diversity of benign data. This limitation indicated that SMOTEN was not fully effective in addressing the dataset imbalance. The test also highlighted that models performed better on malware types with over 1,000 samples, while those with fewer samples exhibited lower performance.

Table 5. Real-world Performance of GBDT Models (Model Robustness Test)

		LightGBM				CatBoost			
		Time-based		Instance-based		Time-based		Instance-based	
		Default	Tuned	Default	Tuned	Default	Tuned	Default	Tuned
Benign	Precision	0.9315	0.9583	0.9487	0.9634	0.9383	0.9059	0.9398	0.9101
	Recall	0.6018	0.6106	0.6549	0.6991	0.6726	0.6814	0.6903	0.7168
	F1-Score	0.7312	0.7459	0.7749	0.8103	0.7835	0.7778	0.7959	0.8020
Malicious	Precision	0.9889	0.9891	0.9904	0.9916	0.9908	0.9911	0.9913	0.9921
	Recall	0.9988	0.9993	0.9990	0.9993	0.9998	0.9980	0.9988	0.9980
	F1-Score	0.9938	0.9942	0.9947	0.9954	0.9948	0.9945	0.9950	0.9950
Macro Avg.	Precision	0.9602	0.9737	0.9695	0.9775	0.9646	0.9485	0.9965	0.9511
	Recall	0.8003	0.8049	0.8269	0.8492	0.8357	0.8397	0.8445	0.8574
	F1-Score	0.8625	0.8701	0.8848	0.9028	0.8891	0.8862	0.8955	0.8985
Weighted Avg.	Precision	0.9873	0.9883	0.9892	0.9908	0.9894	0.9888	0.9899	0.9898
	Recall	0.9879	0.9886	0.9896	0.9910	0.9898	0.9893	0.9903	0.9903

F1-Score	0.9866	0.9874	0.9886	0.9903	0.9890	0.9886	0.9896	0.9897
AUC-ROC	0.8003	0.8049	0.8269	0.8492	0.8357	0.8397	0.8445	0.8574
Overall Accuracy	0.9879	0.9886	0.9896	0.9910	0.9898	0.9893	0.9903	0.9903

Repeated Holdout Test: This test confirmed that the models were overfitted, as indicated by significant differences in the Recall and F1-Score for benign samples when comparing per-fold results to the actual performance. This overfitting was attributed to the dataset's heavy imbalance among malicious samples and the insufficient diversity of benign samples.

McNemar Test: Applying the McNemar test with a null hypothesis of equal performance across models showed no significant differences between models based on the GBDT algorithm, tuning, or behavior type. However, a notable exception was found between Time-based and Instance-based models of the tuned LightGBM, where tuning significantly enhanced the performance of the Instance-based model.

Table 6. Significant Differences of GBDT Models (McNemar Test)

Comparison		p-value	Result
LightGBM vs CatBoost	Default LightGBM TB vs Default CatBoost TB	0.0700	Equal
	Default LightGBM IB vs Default CatBoost IB	0.4386	Equal
	Tuned LightGBM TB vs Tuned CatBoost TB	0.4913	Equal
	Tuned LightGBM IB vs Tuned CatBoost IB	0.4669	Equal
Default vs Tuned	Default LightGBM TB vs Tuned LightGBM TB	0.2568	Equal
	Default LightGBM IB vs Tuned LightGBM IB	0.0578	Equal
	Default CatBoost TB vs Tuned CatBoost TB	0.5930	Equal
	Default CatBoost IB vs Tuned CatBoost IB	1.0000	Equal
Time-based vs Instance-based	Default LightGBM TB vs Default LightGBM IB	0.0895	Equal
	Tuned LightGBM TB vs Tuned LightGBM IB	0.0184	Not Equal
	Default CatBoost TB vs Default CatBoost IB	0.6698	Equal
	Tuned CatBoost TB vs Tuned CatBoost IB	0.4652	Equal

Generalized Model Performance Test: Evaluating the models on a non-biased dataset demonstrated that the GBDT models outperformed traditional machine learning models such as SVM and MLP.

Table 7. Generalized Model Performance of the GBDT Models (Generalized Model Performance Test)

		LGBM	CatBoost	SVM	MLP
Benign	Precision	0.4340	0.0541	0.0338	0.0202
	Recall	0.8553	0.8421	0.8816	0.8816
	F1-Score	0.0827	0.1017	0.0650	0.0395

	Support				76
Malicious	Precision	0.9997	0.9997	0.9998	0.9997
	Recall	0.9625	0.9707	0.9498	0.9148
	F1-Score	0.9807	0.9850	0.9741	0.9554
	Support				38151
Macro Avg.	Precision	0.5216	0.5269	0.5168	0.5100
	Recall	0.9089	0.9064	0.9157	0.8982
	F1-Score	0.5317	0.5433	0.5196	0.4974
Weighted Avg.	Precision	0.9978	0.9978	0.9978	0.9978
	Recall	0.9623	0.9704	0.9496	0.9147
	F1-Score	0.9790	0.9832	0.9723	0.9536
ROC-AUC		0.9089	0.9064	0.9157	0.8982
Overall Accuracy		0.9623	0.9704	0.9496	0.9147

Model Exclusivity Test: This test revealed that the models did not depend on exclusive API Calls for distinguishing between malicious and benign samples. Only 10.41% of the entire dataset utilized API Calls found to be exclusive, suggesting that these features are not universally employed.

Table 8. Validation of Model Reliance to Exclusive API Calls among Benign and Malicious Samples (Exclusivity Test)

Exclusivity Test		LightGBM		CatBoost	
		Original	Exclusivity	Original	Exclusivity
Benign	Precision	0.9315	0.9231	0.9383	0.8333
	Recall	0.6018	0.1062	0.6726	0.1770
	F1-Score	0.7312	0.1905	0.7835	0.2920
Malicious	Precision	0.9889	0.9754	0.9908	0.9773
	Recall	0.9988	0.9998	0.9988	0.9990
	F1-Score	0.9938	0.9874	0.9948	0.9880
Macro Avg.	Precision	0.9602	0.9493	0.9646	0.9053
	Recall	0.8003	0.5530	0.8357	0.5880
	F1-Score	0.8625	0.5890	0.8891	0.6400
Weighted Avg.	Precision	0.9873	0.9740	0.9894	0.9734
	Recall	0.9879	0.9753	0.9898	0.9765
	F1-Score	0.9866	0.9656	0.989	0.9690
AUC-ROC		0.8003	0.5530	0.8357	0.5880
Overall Accuracy		0.9879	0.9753	0.9898	0.9765

Trojan-Undersampling Test: The dominance of Trojan samples slightly impacted the models' ability to detect benign samples. Undersampling these Trojan samples led to a slight performance improvement while maintaining the same general trends observed in the original dataset.

Table 9. Effects of the number of Trojan Samples to Model Performance (Trojan Undersampling Test)

Model Performance		Original Model		Undersampled Model	
		LightGBM	CatBoost	LightGBM	CatBoost
Benign	Precision	0.9315	0.9383	0.7458	0.7664
	Recall	0.6018	0.6726	0.7788	0.7257
	F1-Score	0.7312	0.7835	0.7619	0.7455
	Support				113
Malicious	Precision	0.9889	0.9908	0.9938	0.9923
	Recall	0.9988	0.9998	0.9925	0.9938
	F1-Score	0.9938	0.9948	0.9931	0.993
	Support				4011
Macro Avg.	Precision	0.9602	0.9646	0.8698	0.8793
	Recall	0.8003	0.8357	0.8856	0.8597
	F1-Score	0.8625	0.8891	0.8775	0.8692
Weighted Avg.	Precision	0.9873	0.9894	0.9870	0.9861
	Recall	0.9879	0.9898	0.9867	0.9864
	F1-Score	0.9866	0.9890	0.9868	0.9862
AUC-ROC		0.8003	0.8357	0.9867	0.8597
Overall Accuracy		0.9879	0.9898	0.8856	0.9864

Benign Performance Validation Test: The results suggest that the models' benign sample misclassification is only prevalent in clusters and types that are predominantly Trojan, PUA, Adware, or Downloader, along with Ransomware and Miner due to its behavioral similarities to benign samples. A low misclassification rate was observed for samples from cluster "c_6" due to it containing the greatest number of benign samples from PortableApps, implying that the model automatically classifies the traits from such samples as benign compared to those not sourced from PortableApps. Consequently, high misclassification rates were observed for benign samples from cluster "c_0" and "c_4" due to the prevalence of installmonster and bundler malware families which are akin to the majority of subtypes of benign samples identified on the dataset.

Table 10. Misclassification Rate of the Models on Benign Samples (Benign Performance Validation)

Cluster	Major Families & Types	% of Benign	Default LGBM TB	Default CATB TB
c_0	installmonster & bundler (PUA and Adware)	26.23%	51.11%	54.05%
c_1	ekstrak & fareit (Trojan-based Stealer/Spyware)	1.76%	4.44%	2.70%
c_3	installmonster, bundler, soft32downloader, downloadware (PUA & Adware)	5.38%	4.44%	5.41%
c_4	bundler, istartsurf, chapak, and installmonster (Trojans/Trojan-based, Adware, & PUA)	29.94%	31.11%	32.43%
c_5	msil & razy (Trojan/Trojan-based, Spyware/Backdoor, Virus, Ransomware)	1.11%	4.44%	2.70%
c_6	adload, installmonster, istartsurf, chapak, bundler, nsis, & prepscram (Trojan/Trojan-based Adware, PUA, Downloader)	35.59%	4.44%	2.70%

Table 11. Subtypes of Benign Samples from the Dataset

Clusters	Types of Benign Samples
c_0*, c_4	• Mainly Windows applications/utilities or non-installer PEs • Some installers/uninstallers • Few PortableApps*
c_1, c_3	• Mainly utilities • Some installers/uninstallers
c_5	• Mainly Windows applications/utilities or non-installer PEs • Some installers/uninstallers
c_6	• Notable quantity of PortableApps. • Mainly Windows applications/utilities or non-installer PEs • Some installers/uninstallers.

Model Speed Test: LightGBM consistently outperformed CatBoost in terms of both training and prediction times across various configurations, reinforcing its efficiency advantage.

Table 12. Fit and Prediction Times of GBDT Models

		Fit Time (s)			Predict Time (ns)		
		Best Time	Avg. Time	Worst Time	Best Time	Avg. Time	Worst Time
Default	LGBM TB	1.23	1.41	1.62	36.36	40.98	48.49
	LGBM IB	0.62	0.80	1.46	33.94	37.83	50.91
	CATB TB	28.87	32.07	34.07	429.20	449.11	477.68
	CATB IB	19.43	23.11	23.86	412.21	424.83	453.44

Tuned	LGBM TB	10.29	10.66	10.93	109.11	118.57	140.66
	LGBM IB	18.89	20.20	22.51	196.41	208.78	223.07
	CATB TB	256.10	256.76	257.35	441.35	445.42	467.95
	CATB IB	44.54	44.66	44.73	407.37	419.25	431.62

5 Conclusions and Recommendations

This study successfully addressed its primary objectives, which involved the development and application of Gradient Boosted Decision Tree (GBDT) models for behavior-based malware detection. The research utilized and thoroughly analyzed one of the few publicly available datasets suitable for this purpose, employing various techniques, including string pattern analysis and Principal Component Analysis (PCA) combined with K-Means clustering. The analysis demonstrated that clustering malware by family provides more insightful and conclusive results than clustering by type, as the former reveals clearer contrasts between malicious and benign samples. Additionally, the study highlighted the limitations of relying solely on API Calls as a feature for behavior-based malware detection, particularly in rare cases where malware and benign software share identical API Calls or Call Patterns. The findings emphasize the importance of a well-balanced dataset with an ample and diverse set of samples for high-quality results. The study acknowledges its limitations and suggests that future research could benefit from further investigation into the behavior of both benign and malicious samples by analyzing recurring API Call Patterns, potentially utilizing bioinformatics techniques.

The study also implemented and evaluated the use of GBDT models—LightGBM and CatBoost—in behavior-based malware detection. These models were tested on both Time-based and Instance-based behaviors, and their performance was rigorously evaluated across various metrics. Despite some limitations in certain aspects, the GBDT models proved to be both functional and practical, with their performance generally exceeding the minimum requirements specified. Notably, while the differences between the GBDT models were not statistically significant, they did outperform traditional machine learning models by a small margin.

Based on the findings, several recommendations are proposed for future research. First, it is advisable to consider a broader range of samples, particularly by function or purpose for benign samples and by family for malicious samples. This approach would ensure that even minority classes or types have at least 1,000 samples, thereby enhancing the robustness of the dataset. Second, it is recommended that future studies cover more time units to provide greater context for behavioral analysis, potentially incorporating DLL sequences as supplementary data when API Calls alone are insufficient. Additionally, datasets must be made available through platforms such as Kaggle, IEEE Dataport, or GitHub complete with a transparent and detailed explanation, as its own study, as to where the PE samples were obtained from and what are the specific details about the PE samples used (e.g., application/malware type) prior to extraction of behavioral data. These recommendations aim to refine the methodolo-

gies and datasets used in behavior-based malware detection, ultimately contributing to more accurate and reliable detection systems.

References

1. Anderson, H.S., Roth, P.: EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. (2018). https://doi.org/10.48550/ARXIV.1804.04637.
2. Harang, R., Rudd, E.M.: SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection. (2020). https://doi.org/10.48550/ARXIV.2012.07634.
3. Oliveira, A.: Malware Analysis Datasets: API Call Sequences, https://ieee-dataport.org/open-access/malware-analysis-datasets-api-call-sequences, (2019). https://doi.org/10.21227/TQQM-AQ14.
4. Oliveira, A., Sassi, R.J.: Behavioral Malware Detection Using Deep Graph Convolutional Neural Networks. (2019). https://doi.org/10.36227/techrxiv.10043099.v1.
5. Maniriho, P.: MalbehavD-V1, https://github.com/mpasco/MalbehavD-V1, (2022).
6. Catak, F.O., Yazı, A.F.: A Benchmark API Call Dataset for Windows PE Malware Classification. (2019). https://doi.org/10.48550/ARXIV.1905.01999.
7. Han, W., Xue, J., Wang, Y., Liu, Z., Kong, Z.: MalInsight: A systematic profiling based malware detection framework. J. Netw. Comput. Appl. 125, 236–250 (2019). https://doi.org/10.1016/j.jnca.2018.10.022.
8. Goyal, M., Kumar, R.: Machine Learning for Malware Detection on Balanced and Imbalanced Datasets. In: 2020 International Conference on Decision Aid Sciences and Application (DASA). pp. 867–871 (2020). https://doi.org/10.1109/DASA51403.2020.9317206.
9. Goyal, M., Kumar, R.: The Pipeline Process of Signature-based and Behavior-based Malware Detection. In: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA). pp. 497–502 (2020). https://doi.org/10.1109/ICCCA49541.2020.9250879.
10. Aslan, O., Samet, R.: A Comprehensive Review on Malware Detection Approaches. IEEE Access. 8, 6249–6271 (2020). https://doi.org/10.1109/ACCESS.2019.2963724.
11. Brown, A., Gupta, M., Abdelsalam, M.: Automated Machine Learning for Deep Learning based Malware Detection. (2023). https://doi.org/10.48550/ARXIV.2303.01679.
12. M., G., Sethuraman, S.C.: A comprehensive survey on deep learning based malware detection techniques. Comput. Sci. Rev. 47, 100529 (2023). https://doi.org/10.1016/j.cosrev.2022.100529.
13. Cannarile, A., Dentamaro, V., Galantucci, S., Iannacone, A., Impedovo, D., Pirlo, G.: Comparing Deep Learning and Shallow Learning Techniques for API Calls Malware Prediction: A Study. Appl. Sci. 12, 1645 (2022). https://doi.org/10.3390/app12031645.

14. Zhang, Z.: Microsoft Malware Prediction Using LightGBM Model. In: 2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). pp. 41–44 (2022). https://doi.org/10.1109/ICBAIE56435.2022.9985850.
15. Marais, B., Quertier, T., Morucci, S.: AI-based Malware and Ransomware Detection Models. (2022). https://doi.org/10.48550/ARXIV.2207.02108.
16. Galen, C., Steele, R.: Empirical Measurement of Performance Maintenance of Gradient Boosted Decision Tree Models for Malware Detection. In: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). pp. 193–198 (2021). https://doi.org/10.1109/ICAIIC51459.2021.9415220.