Reverse Engineering of Android Malware Classification using Semi-supervised Learning

SujayKumar Reddy M
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India
sujaykumarreddy.m2020@vitstudent.ac.in

Phanitha Sri Thota
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India
thotaphanitha.sri2020@vitstudent.ac.in

***Abstract*—The escalating prevalence of Android malware poses a substantial threat to online security, necessitating innovative approaches to combat this growing menace. Identifying novel and intricate malware variants remains a formidable challenge in the realm of software security. Existing malware detection methods primarily relying on static features often prove inadequate in countering sophisticated modern malware. While dynamic detection methods hold promise, they frequently fail to leverage the full spectrum of malware characteristics. This paper proposes a novel malware classification framework utilizing semi-supervised learning methods utilizing both unsupervised and supervised learning algorithms. We use the CCCS-CIC-AndMal-2020 dataset encompassing 13 prominent malware categories and 191 eminent malware families. Our results suggested that PCA+XGBoost accomplished by reducing the original dimensionality from 142 features to a more compact set of 48 features. The utilization of PCA in conjunction with the XGBoost model not only enhances the computational efficiency but also preserves the essential information required for classification.**

***Index Terms*—Machine Learning, Security, Reverse Engineering, Malware Analysis**

I. Introduction

The escalating bulk of Android malware poses a substantial threat to online security, necessitating innovative approaches to combat this growing menace. Malicious software adversely affects systems by compromising the integrity, confidentiality, and availability of data on Android devices. This impact includes unauthorized access, data breaches, financial losses, and potential harm to sensitive personal information. Cybersecurity statistics underline the urgency of addressing these threats, with an increase in cyberattacks annually, growing financial losses due to cybercrime, and the rising sophistication of attack vectors. Palo Alto Netork reports [1] suggest a rise in traditional malware techniques exploiting interest in AI/ChatGPT and an increase in vulnerabilities by 55% compared to 2021.

Identifying novel and intricate malware variants remains a formidable challenge in the realm of software security. Existing malware detection methods, primarily relying on static features, often prove inadequate in countering sophisticated modern malware. Reverse engineering and dynamic analysis play crucial roles in this context. Reverse engineering is essential to understanding malware behavior, capabilities, and vulnerabilities. Dynamic analysis, involving the execution of malware within a controlled environment, is necessary to observe real-time behavior, as static analysis alone may prove insufficient. The deceptive nature of Android malware, often distributed through malicious Android Package (APK) files, underscores the need for these techniques.

In malware detection for Android apps, research was conducted by Urooj et al [2], achieving a 96.24% accuracy using machine learning and static features. While notable, limitations are acknowledged in the study, such as a lack of dynamic features, and challenges in sustaining models amid app advancements are highlighted. Megira et al [11], focusing on Android malware. Employing a stacking-based classification approach, the authors analyze host-level encrypted traffic, machine learning-based detection, and PAM clustering for Android malware. The goal is to understand infection mechanisms, assess threat levels, and enhance protection. Leveraging static and dynamic analysis alongside reverse engineering methods like assembly and debugging, the study concludes that combining these techniques yields more accurate results for malware analysis.

This paper contributes to the cybersecurity landscape by proposing an innovative malware classification framework. By addressing the impact of malware on systems, providing relevant cybersecurity statistics, emphasizing the importance of reverse engineering and dynamic analysis, and contextualizing the challenges posed by Android malware and APKs, the paper strives to enhance our understanding and mitigation of the escalating threat of Android malware. The structure of the remaining paper is outlined as follows: Section II reviews related works in the field. Section III details the design of the proposed methodology. Section IV provides an in-depth explanation of the algorithm utilized in our approach. The experiments and corresponding results will be presented in Section V. Section VI addresses research challenges and outlines conclusions for future directions.

II. Related Work

Several works have been done to analyze and detect malware. Burji et al. [3] employed reverse engineering and data mining techniques on the Nugache worm, extracting behavioral patterns from 49 malware samples. Their study advocates for the use of rough set-based machine learning tools to establish distinctive patterns for malware detection. Emphasizing

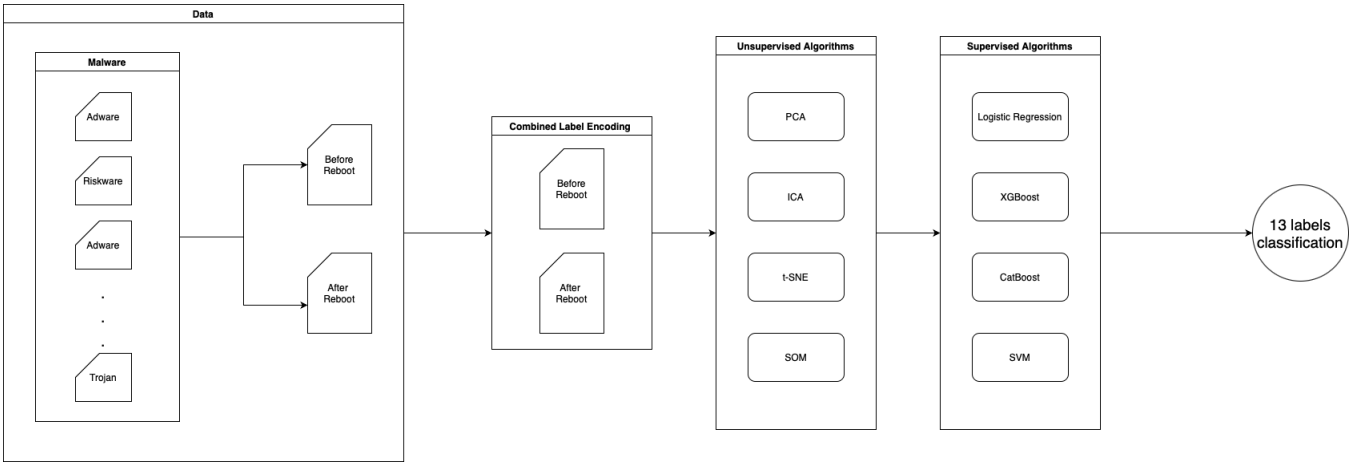


Fig. 1: The Proposed Methodology for the Semi-supervised Algorithms

the effectiveness of these tools, the research underscores the importance of integrating them with current security solutions for practical application in everyday computing scenarios. Jain et al. [4] utilized reverse engineering and SVM for hardware trojan detection in integrated circuits (ICs), achieving superior performance with the Radial kernel. Despite challenges like unknown trojan-free ICs, their study demonstrates SVM's efficacy in identifying trojan-infected ICs, particularly proficient in detecting small trojans with increased accuracy. Tsague et al [5] side-channel analysis for smart card malware code recovery, employing dimensionality reduction (PCA, LDA) and machine learning. Achieving an 87.40% recognition rate, challenges included real-world recognition (62%) and the need for improved tools. That side channel-based reverse engineering is practical, proposing a valuable disassembler for embedded systems. Rathore et al. [6] applied machine learning and deep learning techniques, focusing on opcode frequency as a feature vector. Despite the complexity of the dataset, Random Forest outperformed Deep Neural Network with a remarkable 99.7% accuracy, showcasing the effectiveness of their approach in treating malware analysis as a machine learning problem.

Poudyal et al. [7] developed a reverse engineering framework integrated with feature generation engines and machine learning (ML) to proficiently identify ransomware. Their methodology, validated on a dataset of 302 malware samples, employed reverse engineering and preprocessing techniques to extract features from ransomware and normal binaries. Across various datasets, experiments revealed detection accuracies ranging from 76% to 97% across different supervised machine-learning algorithms. With over 90% accuracy for seven out of eight classifiers tested, their model highlights the significance of static analysis in discerning ransomware characteristics for effective machine learning-based detection. Cappers et al. [8] introduce EventPad, a visual analytics tool for swift malware analysis, demonstrating its efficacy in network traffic analysis and ransomware detection using real-world datasets. Despite challenges like the "cold start" problem, EventPad's integration of data reduction, visualization techniques, and rule-based analysis provides quick insights, supporting rapid and cost-effective malware analysis. Pfeffer et al. [9] introduce MAAGI, employing reverse engineering and semantic analysis for malware lineage determination. Their methodology, validated with 140 malware samples, achieves a precision-recall F-measure of 73% on GitHub data and 92% on malware data, demonstrating promise but necessitating further refinement and evaluation. Nguyen et al. [10] introduce the MARE methodology and M.D. timeline, offering a structured framework for malware analysis. MARE includes phases like Detection, Isolation & Extraction, Behavior Analysis, and Code Analysis & Reverse Engineering, facilitating systematic analysis. Emphasizing its utility in legal and educational contexts, the authors envision its formalization as admissible evidence. The M.D. timeline, comprising six phases from Detection to Malware Inoculation, aims to enhance understanding and defense against malware threats.

III. Proposed Methodology

The proposed methodology, depicted in Fig. 1, employs a semi-supervised learning approach that utilizes unsupervised algorithms for dimensionality reduction followed by supervised learning algorithms for malware classification. The dataset encompasses a thorough dynamic analysis of each malware instance, executed within an emulated environment. For feature extraction, Rahali et al. [12] employed reverse engineering on the .apk files using apktool [13] where the static features were extracted from the *AndroidManifest.xml* file. Keyes et al. [14] explored the Dynamic Feature Extraction and observed the limitations of static analysis in detecting malware triggered by specific runtime environments. We use the dataset but we classify the malware categories. The Decision Tree algorithm demonstrates superior performance with sparse matrices, outperforming both Naive Bayes and Random Forest classification methods. By intentionally pruning Decision Trees to prevent overfitting, we make them easy to understand, possibly revealing new aspects of malware

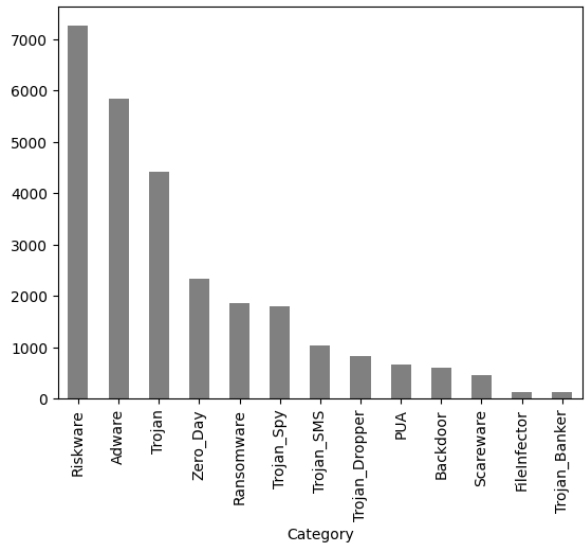


Fig. 2: Malware Categorical Distribution in the Combined Dataset

evolution. This paper proposes an novel semi-supervised algorithm using unsupervised and supevised algorithms. This approach helps to overcome the difficulties posed by sparse matrices in other non-interpretable algorithms like Ensemble based learning methods. Sparse matrices may not function effectively with neural networks, especially when the deep learning frameworks lack optimization for sparse computations. Therefore, we adopt the semi-supervised approach to improve the performance of malware classification.

Given the sparsity of the feature space, unsupervised algorithms like Principle Component Analysis (PCA), Independent Component Analysis (ICA), Self-Organizing Maps (SOM) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are used for dimensionality reduction. These techniques aim to transform the high-dimensional feature space into a lower-dimensional representation while preserving the essential information. Due to the sparse nature of the data, these unsupervised algorithms are often more suitable than neural networks or other algorithms that may not be optimized for such computations. After dimensionality reduction, supervised learning algorithms are applied to classify malware samples. The primary supervised learning algorithms used in this study include logistic regression, XGBoost, CatBoost, and support vector machines (SVMs). These algorithms are selected for their effectiveness in malware classification and their ability to handle high-dimensional data. Extreme hyperparameter tuning is performed to optimize the performance of the supervised learning algorithms. Hyperparameters are the parameters of the learning algorithm that govern the learning process, such as the learning rate and the number of iterations. By tuning these hyperparameters, the performance of the classification models can be significantly improved.

A. Datasets

We utilize the CCCS-CIC-AndMal-2020 dataset [15] for conducting semi-supervised learning on the dynamic analysis dataset of Android malwares. To understand the behavioral variations within different malware categories and families, we extract six categories of features by executing malware within an emulated environment. The primary extracted features encompass API calls, Memory, Network, Battery, Logcat (capturing log messages corresponding to malware functions), and process feature counts. The dataset comprises a comprehensive range of 13 distinct categories of malware families, encompassing Adware, Backdoor, File Infector, Potentially Unwanted Application (PUA), Ransomware, Riskware, Scareware, Trojan, Trojan Banker, Trojan Dropper, Trojan SMS, Trojan Spy, and Zero Day. These categories are observed in the dataset both before and after reboot. The data files were consolidated to form a combined dataset comprising 27,332 rows and 142 columns and the distribution of samples in each category is shown in the Fig. 2.

B. Algorithms Used

Our methodology integrates both unsupervised and supervised algorithms, as outlined in the methodology section. Unsupervised algorithms operate without relying on labeled data. In our approach, we leverage these algorithms primarily for dimensionality reduction purposes. Acknowledging the potential information loss during this process, our aim is to minimize such loss and achieve a compact representation of the data. Specifically, this study employs PCA, ICA, SOM, and t-SNE algorithm for unsupervised dimensionality reduction, aiming to transform data into uncorrelated variables that capture the maximum variance. For classification tasks, we utilize supervised algorithms, including Logistic Regression, XGBoost, CatBoost, and Support Vector Machines (SVM), particularly considering the clustering patterns identified through the unsupervised techniques.

IV. Experimental Results and Discussion

In this section, we delve into the outcomes obtained by our proposed methodology applied to the Android malware dataset. Firstly, we present the results achieved by unsupervised learning algorithms, demonstrating their ability to uncover inherent structures and patterns within the data. Subsequently, we evaluate the performance of supervised learning algorithms, establishing baseline models for our dataset. All experiments were conducted utilizing the comprehensive sklearn-package [16], a robust and widely-adopted machine learning library.

In constructing the dataset, all datasets except ”No_Category” were merged row-wise, excluding instances without manual classification. To encode the categorical variables, label encoding was employed, transforming the 13 class labels into numerical representations ranging from 0 to 12. This dataset was subjected to four supervised learning algorithms and the selection of these algorithms was guided by their distinct working principles and proven

Class (Encoded)	F1 Score for PCA	F1 Score for t-SNE	F1 Score for ICA	F1 Score for SOM
PUA	0.0	0.0	0.0	0.1765
Adware	0.3086	0.4277	0.4103	0.5905
Backdoor	0.0	0.0382	0.0	0.1503
FileInfector	0.0	0.0	0.0	0.0625
Ransomware	0.0	0.4151	0.4928	0.5729
Riskware	0.4173	0.5988	0.5141	0.6824
Scareware	0.0	0.1322	0.3194	0.6267
Trojan	0.2201	0.4023	0.4325	0.5327
Trojan_Banker	0.0	0.0	0.0	0.0
Trojan_Dropper	0.0	0.0614	0.0	0.2517
Trojan_SMS	0.0	0.1511	0.0	0.3289
Trojan_Spy	0.0	0.1727	0.3394	0.6894
Zero_Day	0.0	0.0508	0.0	0.1252
Overall Accuracy	0.29	0.40	0.40	0.56

TABLE I: F1 Scores for Each Class (Logistic Regression)

Class (Encoded)	F1 Score for PCA	F1 Score for t-SNE	F1 Score for ICA	F1 Score for SOM
PUA	0.7881	0.6620	0.7797	0.4192
Adware	0.8314	0.7506	0.8233	0.5990
Backdoor	0.7356	0.6113	0.7444	0.3855
FileInfector	0.56	0.3265	0.5714	0.2128
Ransomware	0.8142	0.7777	0.8110	0.6486
Riskware	0.8584	0.8047	0.8594	0.7132
Scareware	0.7879	0.7042	0.7964	0.4638
Trojan	0.8339	0.7653	0.8337	0.6351
Trojan_Banker	0.5714	0.5283	0.5098	0.1111
Trojan_Dropper	0.6799	0.5978	0.6331	0.4879
Trojan_SMS	0.7093	0.5897	0.7347	0.4344
Trojan_Spy	0.8874	0.8523	0.8825	0.7006
Zero_Day	0.5338	0.4413	0.5479	0.2748
Overall Accuracy	0.81	0.73	0.80	0.60

TABLE II: F1 Scores for Each Class (XGBoost)

Class (Encoded)	F1 Score for PCA	F1 Score for t-SNE	F1 Score for ICA	F1 Score for SOM
PUA	0.7080	0.5397	0.6906	0.3347
Adware	0.7910	0.6911	0.7980	0.5757
Backdoor	0.6527	0.4843	0.6527	0.3333
FileInfector	0.3158	0.1176	0.2632	0.0000
Ransomware	0.8033	0.7346	0.8105	0.6367
Riskware	0.8350	0.7750	0.8386	0.6637
Scareware	0.7788	0.5571	0.8000	0.4505
Trojan	0.8268	0.7211	0.8267	0.5781
Trojan_Banker	0.5455	0.3500	0.6250	0.0000
Trojan_Dropper	0.6528	0.5209	0.6726	0.4518
Trojan_SMS	0.6667	0.4253	0.6860	0.3392
Trojan_Spy	0.8930	0.7922	0.8957	0.6929
Zero_Day	0.5317	0.3673	0.5301	0.2693
Overall Accuracy	0.79	0.68	0.79	0.57

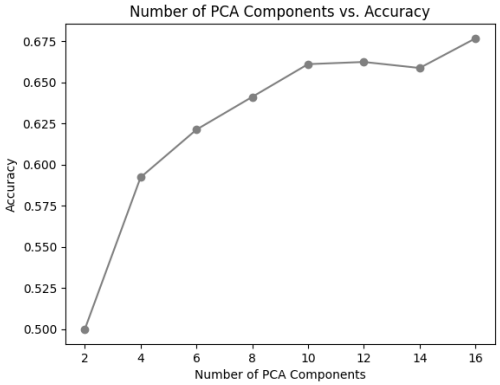
TABLE III: F1 Scores for Each Class (CatBoost)

ability to accurately classify data, as demonstrated in prior research. Among these algorithms, XGBoost, an ensemble learning algorithm, emerged as the most capable, achieving a remarkable accuracy rate of 76% (0.76). Given its superior performance, XGBoost was adopted as the benchmark metric for subsequent exploration into dimensionality reduction techniques.

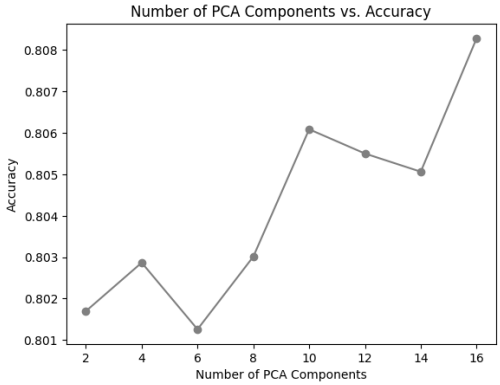
Fig. 3, illustrates the convergence graphs pertaining to the Unsupervised learning methods for Principal Component Analysis (PCA), elucidating the relationships with the associated hyperparameters. Through our analysis, we observed that employing PCA directly onto the algorithm resulted in lower accuracy and greater loss of information. In response to this finding, we introduce a novel feature-wise dimensionality reduction technique designed to incrementally enhance accuracy on a per-feature basis. Our proposed approach involves partitioning the entire dataset vertically into three distinct categories: Memory features, API features, and Other features. Subsequently, we apply three distinct unsupervised algorithms for dimensionality reduction tailored to each feature category. This innovative methodology aims to capitalize on the unique characteristics of each feature subset, fostering a more nuanced and accurate representation of the underlying patterns within the data. Through this nuanced approach, we anticipate achieving improved accuracy and information retention compared to conventional techniques, ultimately enhancing the overall performance of the XGBoost classification model. PCA emerges as the most effective dimensionality reduction technique,

Class (Encoded)	F1 Score for PCA	F1 Score for t-SNE	F1 Score for ICA	F1 Score for SOM
PUA	0.0	0.4564	0.4262	0.0
Adware	0.4732	0.5746	0.6384	0.3831
Backdoor	0.0	0.1078	0.2373	0.0
FileInfector	0.0	0.0	0.0	0.0
Ransomware	0.5659	0.6764	0.6925	0.4181
Riskware	0.6514	0.7076	0.7482	0.4897
Scareware	0.4568	0.4222	0.6533	0.3955
Trojan	0.5238	0.6175	0.6808	0.3356
Trojan_Banker	0.0	0.0	0.0	0.0
Trojan_Dropper	0.1416	0.3776	0.5153	0.0
Trojan_SMS	0.1286	0.3732	0.3835	0.0
Trojan_Spy	0.6882	0.6124	0.7702	0.2321
Zero_Day	0.1064	0.1493	0.2914	0.0713
Overall Accuracy	0.51	0.58	0.65	0.38

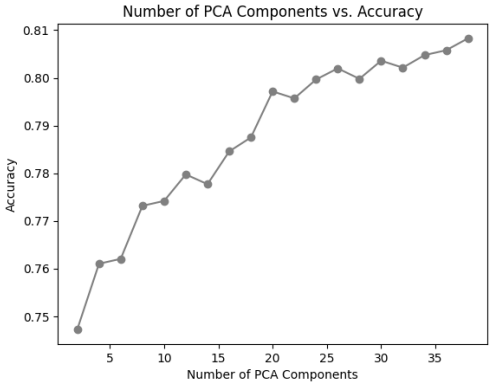
TABLE IV: F1 Scores for Each Class (SVM - Radial Basis kernel)



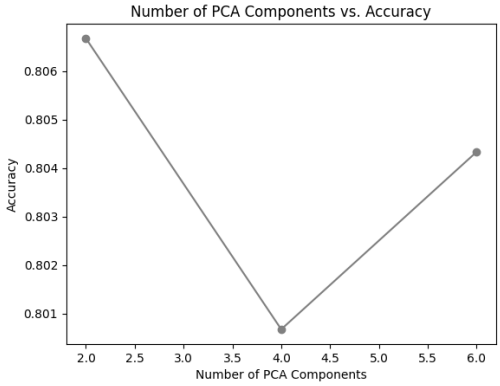
(a) PCA with whole features (142)



(b) PCA with only Memory features



(c) PCA with only API features



(d) PCA with other features

Fig. 3: Dimensionality Reduction Convergence graphs with XGBoost Algorithm

achieving the highest accuracy of 81% when integrated with the XGBoost model. This notable success is accomplished by reducing the original dimensionality from 142 features to a more compact set of 48 features. The utilization of PCA in conjunction with the XGBoost model not only enhances the computational efficiency but also preserves the essential information required for accurate classification. This result underscores the efficacy of PCA in optimizing the performance of machine learning models, particularly when employed in tandem with sophisticated algorithms like XGBoost.

In the subsequent phase of our analysis, we applied supervised learning algorithms to our finely tuned dimensionality reduction models. Tables 1, 2, 3, and 4 present a comprehensive breakdown of class-wise results attained by the four supervised algorithms, with the boosting ensemble learning model demonstrating the highest accuracy. This observation reinforces the efficacy of ensemble learning, specifically boosting, in enhancing classification performance across diverse classes. Additionally, Fig. 4, provides a visual representation of the confusion matrix for each class obtained through the semi-supervised algorithm, offering insights into the model's performance and potential areas for refinement.

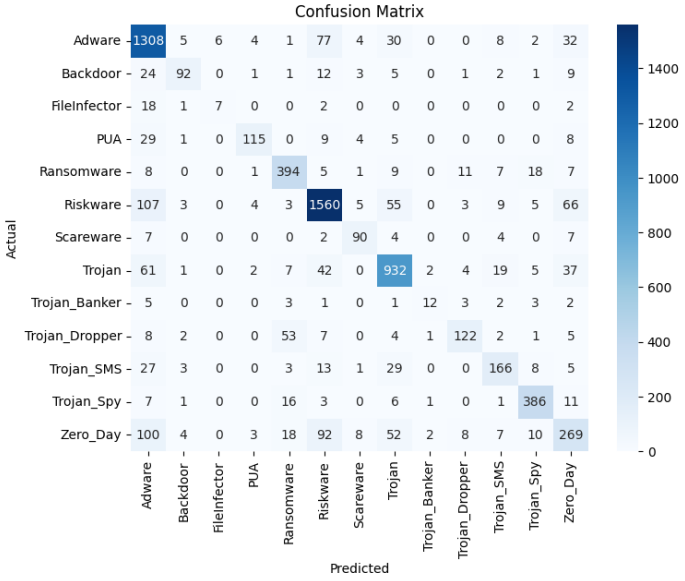


Fig. 4: Confusion matrix for the Semi-supervised PCA+XGBoost model

V. Conclusion and Future work

The landscape of malware detection continues to evolve as researchers grapple with the challenges posed by vast and intricate dimensional datasets. The CCCS-CIC-AndMal-2020 Reverse Engineered static Android malware dataset stands out with nearly 9000 features, a complexity that proves challenging for traditional models such as XGBoost and Logistic Regression. In contrast, Deep Neural Networks and Transformer models exhibit promising capabilities in handling such high-dimensional data. Harnessing the full potential of advanced neural network architectures such as those incorporating self-attention layers and transformers necessitates comprehensive fine-tuning, particularly in the context of multi-label classification tasks. This fine-tuning process is essential to optimize the model's performance and ensure its effectiveness in accurately classifying malware samples. The intricacies of understanding and effectively utilizing models with high intent underscore the ongoing need for sophisticated approaches in the field of malware detection research.We plan to integrate self-attention layers and transformers into our neural network architectures, aiming to elevate the accuracy of malware classification.

References

[1] *2023 Unit 42 Network Threat Trends Research Report.*, https://start.paloaltonetworks.com/unit-42-network-threat-trends-report-malware-2023.html [Accessed : 01-Dec-2023]

[2] Urooj, Beenish, et al. "Malware detection: a framework for reverse engineered android applications through machine learning algorithms." IEEE Access 10 (2022): 89031-89050.

[3] Burji, Supreeth, Kathy J. Liszka, and C-C. Chan. "Malware analysis using reverse engineering and data mining tools." 2010 International Conference on System Science and Engineering. IEEE, 2010.

[4] Jain, Girishma, Sandeep Raghuwanshi, and Gagan Vishwakarma. "Hardware Trojan: Malware detection using reverse engineering and SVM." Intelligent Systems Design and Applications: 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017) held in Delhi, India, December 14-16, 2017. Springer International Publishing, 2018.

[5] Tsague, Hippolyte Djonon, and Bheki Twala. "Reverse engineering smart card malware using side channel analysis with machine learning techniques." 2016 IEEE International Conference on Big Data (Big Data). IEEE, 2016.

[6] Rathore, Hemant, et al. "Malware detection using machine learning and deep learning." Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6. Springer International Publishing, 2018.

[7] Poudyal, Subash, Kul Prasad Subedi, and Dipankar Dasgupta. "A framework for analyzing ransomware using machine learning." 2018 IEEE symposium series on computational intelligence (SSCI). IEEE, 2018.

[8] Cappers, Bram CM, et al. "Eventpad: Rapid malware analysis and reverse engineering using visual analytics." 2018 IEEE Symposium on Visualization for Cyber Security (VizSec). IEEE, 2018.

[9] Pfeffer, Avi, et al. "Malware analysis and attribution using genetic information." 2012 7th International Conference on Malicious and Unwanted Software. IEEE, 2012.

[10] Nguyen, Cory Q., and James E. Goldman. "Malware analysis reverse engineering (MARE) methodology & malware defense (MD) timeline." 2010 Information Security Curriculum Development Conference. 2010.

[11] Megira, S., A. R. Pangesti, and F. W. Wibowo. "Malware analysis and detection using reverse engineering technique." Journal of Physics: Conference Series. Vol. 1140. No. 1. IOP Publishing, 2018.

[12] Rahali, Abir, et al. "Didroid: Android malware classification and characterization using deep image learning." 2020 The 10th international conference on communication and network security. 2020.

[13] Ibotpeaches,A tool for reverse engineering Android apk files, 2019. [Online]. Available: https://ibotpeaches.github.io/Apktool/. [Accessed: 01-Dec-2023].

[14] Keyes, D. S., Li, B., Kaur, G., Lashkari, A. H., Gagnon, F., & Massicotte, F. (2021, May). EntropLyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics. In 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS) (pp. 1-12). IEEE.

[15] Canadian Institute for Cybersecurity (CIC). CCCS-CIC-AndMal-2020. Canadian Institute for Cybersecurity (CIC) Project in Collaboration with Canadian Centre for Cyber Security (CCCS); Canadian Institute for Cybersecurity (CIC): Fredericton, NB, Canada, 2020; Available online: https://www.unb.ca/cic/datasets/andmal2020.html (accessed on 9 April 2023).

[16] Pedregosa, Fabian. "Scikit-learn: Machine learning in python Fabian." Journal of machine learning research 12 (2011): 2825.