Efficient Malware Classification Using Multiprocessing and Bag-of-Words Vectorization*

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—The increasing prevalence of malware necessitates efficient and accurate classification techniques to enhance cybersecurity measures. This study presents a novel approach to malware classification leveraging multiprocessing and Bag-of-Words (BoW) vectorization. We employ a balanced subset of the Microsoft Malware Classification dataset, extracting hexadecimal strings from malware binaries and transforming them into feature vectors using a custom HexVectorizer. To address the computational intensity, we parallelize the vectorization process using Python's multiprocessing library. Our classification model, based on XGBoost's 'XGBClassifier', demonstrates high accuracy and efficiency, underscoring the potential of our approach in real-time malware detection and classification systems. This paper details the methodology, implementation, and performance evaluation, providing a comprehensive solution for large-scale malware classification.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Introduction

II. Related Work

Malware classification has been a significant area of research within the cybersecurity domain due to the ever-increasing threats posed by malicious software. Numerous approaches have been developed to classify malware efficiently, ranging from traditional machine learning techniques to advanced deep learning models.

One of the earlier works in this area is by Nataraj et al. (2011), who proposed a method for visualizing malware binaries as grayscale images and used image processing techniques for automatic classification. This approach demonstrated that visual features could be effective for identifying different malware families, providing a novel perspective on malware classification .

Kolosnjaji et al. (2016) explored the use of deep learning for classifying malware based on system call sequences. They employed convolutional neural networks (CNNs) to capture the patterns in system call traces, achieving significant improvements over traditional machine learning methods. Their work highlighted the potential of deep learning techniques in handling the complexity and variability of malware behavior.

Raff et al. (2018) introduced a distinctive approach where they trained a deep learning model by "eating" the entire executable file, using a recurrent neural network (RNN) architecture. This method eliminated the need for feature engineering by allowing the model to learn directly from the raw binary data. Their results showed that the model could achieve high accuracy in malware detection and classification, demonstrating the effectiveness of end-to-end learning approaches .

Vinayakumar et al. (2017) conducted a comprehensive evaluation of shallow and deep neural networks for ransomware detection and classification. They compared different network architectures and feature sets, concluding that deep learning models, particularly those incorporating temporal and spatial features, outperformed traditional machine learning classifiers. Their research provided insights into the optimal configurations for neural networks in malware classification tasks .

More recent work by Verma (2023) on Kaggle highlighted the practical aspects of implementing custom packages within kernels for efficient malware classification. Verma's contributions emphasize the importance of modular and scalable solutions in handling large-scale datasets, which is crucial for real-world applications of malware detection systems .

Our approach builds on these foundational works by incorporating multiprocessing techniques to enhance the efficiency of feature extraction and vectorization processes. By leveraging the computational power of modern processors, we aim to address the scalability issues associated with large malware datasets. Additionally, we utilize the XGBoost 'XGBClassifier', known for its robust performance and scalability, to achieve high accuracy in malware classification. Our methodology combines the strengths of feature engineering, parallel processing, and ensemble learning to provide a comprehensive solution for malware detection and classification.

III. Dataset Description

The dataset used in this study is derived from the Microsoft Malware Classification Challenge (BIG 2015) available on Kaggle. This dataset provides a comprehensive collection of malware samples, encompassing a variety of malware families, which is instrumental in developing and evaluating malware classification models.

A. Structure and Composition

The dataset consists of two main components for each malware sample:

- **Byte Files ('.bytes' files):** These files contain the raw hexadecimal representation of the binary code of the malware samples. Each line in a '.bytes' file consists of an address in memory followed by a series of hexadecimal byte values. These files are essential for extracting features that represent the behavior and characteristics of the malware.
- **Manifest Files ('.asm' files):** These files contain metadata and assembly language instructions of the malware binaries. They provide additional context and features that could be used for deeper analysis, although our current approach primarily focuses on the hexadecimal content.

B. Malwrae Families

The dataset includes malware samples from nine distinct families, each represented by a unique label. These families are:

- **Ramnit:** A family of malware known for infecting executable files and HTML files, as well as stealing sensitive information.
- **Lollipop:** A type of Adware that displays intrusive advertisements and can potentially download other malicious programs.
- **Kelihos ver3:** The third version of the Kelihos botnet, which is involved in sending spam emails, stealing personal data, and other malicious activities.
- **Vundo:** A type of Trojan horse known for causing pop-up advertisements and compromising system security.
- **Simda:** A backdoor Trojan that allows remote access to the infected system, often used for distributing other malware.
- **Tracur:** A Trojan that redirects web traffic to malicious sites and steals personal information.
- **Kelihos ver1:** The first version of the Kelihos botnet, with similar functionalities as version 3 but with different signatures and behaviors.
- **Obfuscator.ACY:** Also known as Stegoloader, this malware is known for using steganography to hide its presence and steal information from infected systems.
- **Gatak:** Also known as Stegoloader, this malware is known for using steganography to hide its presence and steal information from infected systems.

C. Data Preprocessing

The dataset includes malware samples from nine distinct families, each represented by a unique label. These families are:

- **Hexadecimal String Extraction:** From the **bytes** files, the hexadecimal strings are extracted while removing non-hexadecimal characters and memory addresses. This results in a clean sequence of hex values representing the malware binary.
- **Sampling and Balancing:** The dataset is sampled to create a balanced subset, ensuring that each malware family is represented equally. This step is crucial to prevent the model from being biased towards more prevalent families in the dataset.
- **Feature Vectorization:** Using a custom **HexVectorizer**, the hexadecimal strings are converted into feature vectors using the Bag-of-Words (BoW) approach. This involves tokenizing the hex values and creating a sparse matrix representation of the malware samples.
- **Inclusion of File Size:** In addition to the BoW vectors, the file size of each malware sample is included as an additional feature. This helps capture the differences in file size across different malware families, potentially improving classification accuracy.

IV. Methodology

V. Model Implementation

The model implementation for malware classification in this study involves several key steps, from feature extraction to model training and evaluation. Here, we detail the process of implementing the XGBoost **XGBClassifier** and optimizing it for efficient and accurate classification of malware samples.

A. Feature Extraction and Vectorization

VI. Results and Discussion

A. Conclusion

B. Figures and Tables

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

a) Positioning Figures and Tables:

Fig. 1. Example of a figure caption.

ACKNOWLEDGMENT

REFERENCES

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.