Behavioural Analysis of Malware by Selecting Influential API Through TF-IDF API Embeddings

Binayak Panda[1], Sudhanshu Shekhar Bisoyi[2], Sidhanta Panigrahy[3]
Dept. of Computer Science and Engineering-Institute of Technical Education and Research,
Siksha 'O' Anusandhan (Deemed to be) University Bhubaneswar, Odisha, India[1]
Dept. of Computer Science and Information Technology-Institute of Technical Education and Research,
Siksha 'O' Anusandhan (Deemed to be) University, Bhubaneswar, Odisha, India[2]
Haas School of Business, University of California, Berkeley, CA, United States[3]

***Abstract*—The constant threat of malware makes studying its behavior an ongoing task. Malware identification and classification challenges can be solved better by analyzing software behaviorally rather than using conventional hashcode-based signatures. API sequence represents the behavior of any program when collected during its execution. Considering API sequences gathered while the malware was being executed in controlled conditions, this report addresses the issue of choosing influential APIs for malware. The suggested feature selection method *SelectAPI* in this research selects key features, i.e., significant APIs, that can better classify malware using TF-IDF API embeddings. Two machine learning models, Random Forest, which ensemble several estimators implicitly, and Support Vector Classifier, a standard non-linear model, are trained and evaluated to validate the importance of the chosen APIs. The proposed API selection methodology, called SelectAPI, has shown promising results. It achieves accuracy, macro-avg precision-score, macro-avg recall-score, and macro-avg F1-score of 0.76, 0.77, 0.76, and 0.76, respectively. This method focuses on selecting influential APIs and has resulted in significantly improved performance on the open-benchmark multiclass dynamic-API-Sequence based malware dataset, MAL-API-2019. These results surpass the previously best-known accuracy value of 0.60 and reported F_1-Score of 0.61.**

***Keywords*—Malware analysis; behavioural analysis; API sequence; multiclass malware; TF-IDF; API embeddings**

I. Introduction

Every person uses a variety of devices and apps over the internet to fulfil their everyday requirements related to banking, e-commerce, and many others. The most significant threat to these devices and apps is malware, a computer software. Malware is designed to perform various detrimental actions on the devices and applications of its victims. Attachments in electronic mails, ads, potentially unwanted softwares, and open utility applications are some ways the malware reaches a compromised device or application. The annual threat report for the FY-2022-23 [1] published by Quick Heal reveals that more than 163 million instances of new and known malware were identified in 2021-22. The identified malware samples are of the following families: Trojan, Infector, Ransomware, Cryptojacking, Potentially Unwanted Application (PUA), Adware, and Worm. The detection of malware with respect to their family for 2021 and 2022 are displayed in Fig. 1. Although malware detection is a computationally hard problem, undetected propagation can be limited by applying statistical techniques [2]. Malware analysis is done in two basic ways: static and dynamic. Static analysis involves examining some of the malware's essential characteristics, such as opcode sequences, readable strings, etc., without running the malware. On the other hand, dynamic analysis, also known as behavioural analysis, allows the virus to run in a controlled environment while gathering execution time information such as system call graphs, API sequences, registry file contents, etc. It has been observed that the obfuscation process of generating malicious code is not more effective in dynamic API call analysis in contrast to static type [3]. Researchers continually seek new and improved behavioral analysis methods to identify and categorize malware.

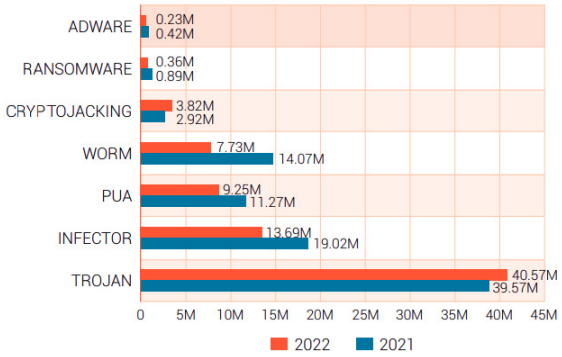

Fig. 1. Quick-heal threat report FY-2022-23 [1].

In this work, the focus has been given to the preprocessing of API sequences to classify malware concerning their families. The open dataset MAl-API-2019 released by Catak [4] is used to train and test machine learning models to demonstrate the preprocessing technique's efficacy in the multiclass malware classification problem. The objectives addressed in this work are:

- Using TF-IDF weight vectors to select influential API as a feature selection technique in the API sequences.
- Ensuring the improvements made using the above feature selection method by considering SVC as a standard non-linear machine learning model and RF as one of the implicit ensemble machine learning models.

Section II of this article lists the literature on malware classification. Section III covers the description of the dataset and the proposed feature selection method, followed by the training

of the aforementioned machine learning models. Section IV lists the findings of the experiment. The final portion highlights the conclusion.

II. Related Work

Malware detection techniques that utilize machine learning have seen significant advancements in recent years. The three fundamental principles, confidentiality, integrity, and availability, of computer security are compromised by malware, which impacts the computer system. By taking advantage of the system's flaws, they get into the computer system without the user or administrator noticing. Malware classification has been the subject of extensive investigation. Various techniques and characteristics are used to categorize new malware into well-known malware categories, identify outliers, and accurately evaluate such abnormalities. The API call sequence is widely used in malware detection techniques to represent the behaviour of malware accurately. Ye et al. gathered the set of API calls from Portable Executables (PE) files for generating the set of feature that was verifiable and comprehensible. A classifier has been trained using these features to identify unknown malware [5]. Geng et al. [6] provided a thorough overview of obfuscated malware and developments in obfuscation techniques, outlining a strategy-based principle from the viewpoint of malcoders. With an emphasis on Windows malware, the authors reviewed a variety of evasion approaches and demonstrated how evasion techniques might be combined to create malware with potent self-defense capabilities. In an experiment comparing adversarial malware generators, Louthánová et al. [7] showed that a combination of methods can efficiently produce new instances that escape detection and automate the generation of malicious activity works more more effectively against detection models that are different from the ones that produced them.

Kong and Yan integrated several malware properties, including opcodes, registers, and API calls, to categorize malware into 11 families. They used pairwise graph matching, ensembled classification, and discriminant distance metric learning to create an efficient system that could identify samples that had not been detected before [8]. In order to remove infrequent elements from the API sequences, Ding et al. proposed an association mining technique based on APIs. To improve detection accuracy, they chose and applied association rules with strong classification capabilities [9].

Recent research on malware analysis has shown that supervised and unsupervised machine learning approaches and deep neural networks are widely used to identify malicious activity with better efficiency, accuracy, and a low rate of false positives. Feature extraction and automatic detection are the most common methods used for malware detection with the help of machine learning [10].

Machine learning methods, such as LR, RF, SVM, KNN, etc., are often used to find and classify unknown samples of different malware families because of their scalability, speed, and flexibility. Han et al. [11] studied the behavior and characteristics of the malicious API call sequence. The analysis reveals a significant correlation between the static and dynamic API calls of the malicious applications. The authors have suggested a model known as MalDAE that can explain the detection of malicious activity by extracting the sequence of API calls from the PE files and cuckoo sandbox, which correlates the dynamic and static API call sequences into a hybrid sequence via semantic mapping. For the most common malware, it can offer a clear explanation and predictive assistance. They outperformed the previous studies with detection and classification accuracy of 97.89% and 94.39%, respectively.

Panda et al. [12] have proposed a host-specific in-memory detection system for malicious programs or software with the help of the TF-IDF API embedding method, particularly on the Windows API call sequences. The authors have prepared a knowledge base for the trusted application and their corresponding behaviour. The cross-validation technique predicts the class of trusted or untrusted applications in the host-specific systems. Mathew and Kumara [13] utilized N-grams and TF-IDF to extract and select features for their research. They proposed an LSTM model for the binary classification of applications as either benign or malicious based on API call sequences. The authors achieved an impressive accuracy score of 0.92 when evaluating the model on previously unseen test API call sequences. Huda et al. [14] have developed a hybrid framework for detecting malicious programs that combines SVM techniques with heuristics derived from the Maximum Relevance and Minimum Redundancy (Mr-MR) filter. This approach employs statistics from API call sequences as feature vectors. The method effectively integrates the ranking score from the filter into the wrapper's selection process. Ultimately, it leverages the strengths of the wrapper, the filter, and the sequences of API calls to efficiently identify malicious activities.

A harmful program may be obfuscated as a new program wrongly classified as benign while maintaining the original behavior and its effects. It may be easy to circumvent the detection procedure for this new program [15].

An ensemble model was presented by Panda et al. [16] for the classification of the imbalanced multiclass malware dataset known as MAL-API-2019. It has been investigated how the API calls relate to one another through the API sequences. They prepared the feature vector for the 1D-CNN using the Skip-gram approach of Word2Vec embedding model. This 1D-CNN model is trained for every class using the one-vs-rest (OvR) technique. To increase classification accuracy, they suggested an ensemble model using ModifiedSoftVoting, a unique soft-voting technique that combines all the class-wise classifiers. The MAL-API-2019 dataset is also used in the training to classify the multiclass malware using RF, DT, SVM, KNN, two-layer LSTM, and single-layer LSTM. They employed single-layer LSTM and obtained a recall and precision of 0.47 compared to all other models [4].

Li and Zheng classified malware types utilizing long-sequence API calls using the GRU and LSTM with the multiclass dataset MAL-API-2019. In LSTM and GRU, the achieved precision is 0.56, and recall is 0.58 and 0.59, respectively [17]. Demirkiran et al. have used a transformer based model with a single layer of transformer block for the classification of malware families. It is found that the suggested transformer-based RTF model outperforms when tested on four benchmark datasets (MAL-API-2019, Olivera, VirusShare, and VirusSample), had F_1-scores-0.61, 0.51, 0.56, and 0.59 and

AUC scores-0.88, 0.83, 0.83, and 0.87 for the three models they compared: Transformer, CANINE-S, and BERT [18].

Gali et al. [19] have combined an AI-based malware detection procedure with the eXplainable Artificial Intelligence (XAI) technology. With precision-54.89%, recall-53.99%, F_1-score-54.31% and accuracy-52.88%, the authors of study [20] obtained the best classification result by evaluating multiple deep-learning models with standard imbalanced multiclass dataset MAL-API-2019 using Binary-LSTM. Quan et al. proposed CAFTrans, a framework that uses CNN and LSTM networks to parse API sequences [20]. When the framework was tested on the MAL-API-2019 dataset, the F1 score was 0.65. They claimed that CAFTrans increases accuracy by detecting malware threats in their respective family more precisely than in other families using the same dataset. By linking Advanced Persistent Threat (APT) malware to the threat actors responsible for it, such as APT groups, Ahmad et al. [21] have improved the analysis of APT malware. APT malware is a serious threat, and averting cyber mishaps requires an awareness of the adversaries behind these attacks. This technique helps cybersecurity researchers and professionals with actionable insight by connecting APT software to threat actors for further analysis of the malware.

Multiclass malware classification problems suffer from class imbalance issues and feature imbalance issues. Without disturbing the class-wise samples, the feature imbalance problem concerning API sequences can be reduced using various techniques. This work applies the TF-IDF word embedding method to API sequences to identify influential APIs and better classify malware to their family.

III. Methodology

API sequences are considered the most important feature in the dynamic analysis of malware. The proposed framework in Fig. 2 illustrates the selection of influential or critical APIs having significance in API sequences concerning the multiclass malware classification problem. This study finds influential APIs by calculating the TF-IDF API embedding for each distinct API during the preprocessing of API sequences to identify malware based on their families. The effectiveness of the API sequences consisting of influential APIs and the efficacy of the models through the proposed feature selection technique is tested using the open dataset MAl-API-2019 published by Catak [4].

A. Dataset Description and Preprocessing

Mal-API-2019 is a multiclass malware dataset on dynamically collected API sequences of eight different classes of malware. This highly imbalanced dataset contains variable-length API sequence records for 7107 pieces of malware from eight different classes. A multiclass dataset is either imbalanced vertically or horizontally. In a vertically imbalanced dataset, the number of samples for each class varies significantly, as depicted in Fig. 3. Meanwhile, in a horizontally imbalanced dataset, the number of features in each sample varies significantly, as depicted in Fig. 4. This approach mitigates the horizontal imbalance issue during feature selection, or API selection, thereby preserving the vertical imbalance issue.

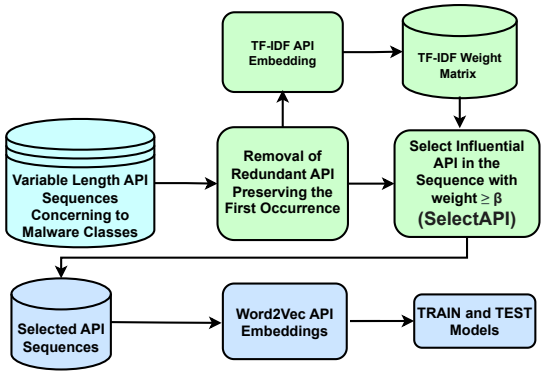

Fig. 2. Proposed framework to select influential APIs in API sequences.

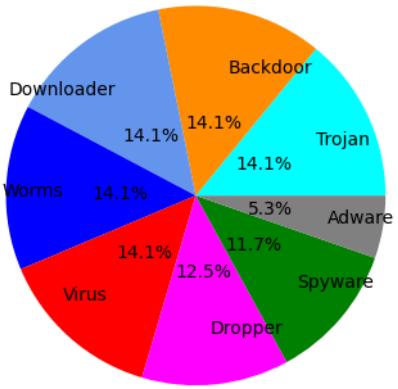

Fig. 3. Vertically imbalanced malware classes.

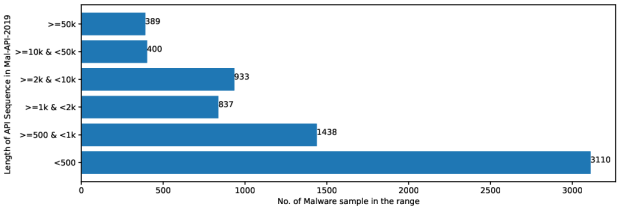

Fig. 4. Horizontally imbalanced API sequences.

Algorithm 1 PreserveFirstAPI

Require: API_{Seq} (API Sequence)
Ensure: $RAPI_{Seq}$ (Reduced API Sequence)
1: $API_{Dict} = \{\ \}$ ▷ Dictionary to track distinct API
2: $RAPI_{Seq} = \phi$
3: **for each** $API \in API_{Seq}$ **do**
4: **if** $API_{Dict}.hashKey(API)! = True$ **then**
5: $RAPI_{Seq}.append(API)$
6: $API_{Dict}[API] = True$
7: **end if**
8: **end for**
9: return $RAPI_{Seq}$

Repeated API calls in the API call sequence evade the malware's potential detection, making it a major runtime

behavior. Every API request performs a distinct machine-level function. The sequence of malware's machine-level tasks is described by maintaining the first call of each unique API in the provided API sequence. By removing duplicate API calls, the algorithm PreserveFirstAPI outlines how to protect each unique API call's initial occurrence. To demonstrate how PreserveFirstAPI works, consider the encoded API sequence with repeated API calls be [A, A, A, C, C, A, A, C, K, K, A, A, C, A, C, K, K, D, D, A, A, A, C, K, K, K, K, D, D, D, T, T, A, A, D, D, A, A, A, C, C, K, K, K, D, D, D, T, K, K, D, T, K, A]. After removing the redundant API by preserving the first occurrence, the encoded API record becomes [A, C, K, D, T]. For each API_{Seq} in the dataset, the Algorithm PreserveFirstAPI finds the reduced API sequence as $RAPI_{Seq}$. The reduced API-sequence dataset is further used by SelectAPI as outlined in Algorithm 2 for selecting influential, i.e., critical APIs having semantic significance over others from each reduced API sequence.

B. Feature Selection

Following the application of PreserveFirstAPI as described in Algorithm 1, TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical technique that guarantees the calculation of the weightage of a distinct-API (i.e., a word) to an API-call-sequence (i.e., a document) in the collection of sequences of API-calls in the dataset (i.e., the collection of documents). In collecting API sequences, TF-IDF reduces the influence of very frequent APIs, which are empirically less informative than less frequent APIs. As mentioned in Eq. (1), TF-IDF assigns weight to an API by multiplying the API's term frequency (TF) with its inverse document frequency (IDF). TF of an API is calculated as mentioned in Eq. (2) considering the number of times the API appears in an API sequence compared to the total number of APIs in the API sequence. The IDF of an API reflects the proportion of API sequences in the dataset that contain the API and is calculated as mentioned in Eq. (3).

$$\textit{TF-IDF}_{API} = \textit{TF}_{API} \times \textit{IDF}_{API} \tag{1}$$

$$TF_{API} = \left(\frac{\textit{Number of occurrences of the API in the API Sequence}}{\textit{Total number of APIs in the API sequence}} \right) \tag{2}$$

$$IDF_{API} = log \left(\frac{\textit{Total number of API sequences in the dataset}}{\textit{Number of API sequences in the dataset contain the API} + 1} \right) \tag{3}$$

The TF-IDF weight matrix for APIs in API sequences of the reduced dataset from PreserveFirstAPI plays a significant role in selecting influential APIs. The steps of SelectAPI outlined in Algorithm 2 say the selection of influential APIs in each API sequence of the updated dataset. A threshold weight β will be decided during experimentation to select influential APIs. During the feature selection, not to disturb the class size distribution of the dataset, an API sequence gets restored to its original form when none of the APIs in the sequence qualifies β. Using the chosen API sequences from SelectAPI, the Word2vec [22] model is used to generate vector representations of various APIs. These vectors capture information about the API's semantics based on the surrounding APIs in the API sequences. These API vectors are further used to train and evaluate machine learning models.

Algorithm 2 SelectAPI

Require: $RAPI_{Seq}$ (The Reduced API Sequence) , $TF - IDF_{API}$ weight matrix and Weight threshold β to select influential API
Ensure: $SelAPI_{Seq}$ (API Sequence of APIs qualifying β)

```
SelAPI_Seq = φ
for each API ∈ RAPI_Seq do
    if TF − IDF_API ≥ β then
        SelAPI_Seq.append(API)
    end if
end for
if Number of terms in SelAPI_Seq = 0 then
    SelAPI_Seq = API_Seq          ▷ Restore the API-Seq
end if                            ▷ If all APIs are deselected
return SelAPI_Seq
```

IV. Experimental Setup and Results

An eight-core "Intel-Corei5-1035G1-CPU @ 1.00GHz " personal computer equipped with 16 Gigabytes of RAM is used during the research to conduct experiments. The computer runs the operating system "Ubuntu-22.04-LTS" and is installed with Anaconda, which has a kernel of Python-3.9 and Jupyter Notebook to conduct experiments. Using the benchmark imbalanced multiclass malware dataset, MAL-API-2019 [4], two machine learning models, SVC, a non-linear model and RF, an implicit ensemble model, are trained and assessed to make sure the proposed feature selection method is effective in classifying malware into the appropriate classes.

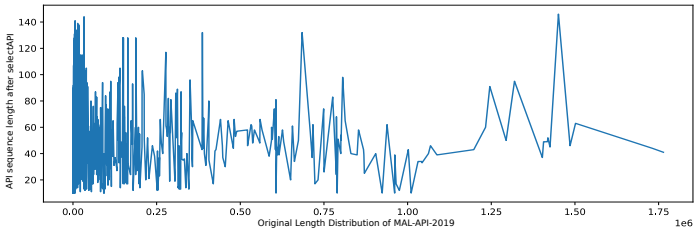

Fig. 5. API-Sequence length variation by SelectAPI with $\beta = 0.13$.

To mitigate the dataset problem concerning API sequence length as depicted in Fig. 4, the suggested feature selection method SelectAPI considered several β values during the experiment to remove insignificant APIs from API sequences. Fig. 5 illustrates the change in API sequence length using SelectAPI against its original length with the best-found value of β as 0.13.

Word2vec [22] model is used with (window size: '10', minimum count: '1', Skip-Gram Selector: '1', and vector size: '100') to generate API embeddings of all the significant

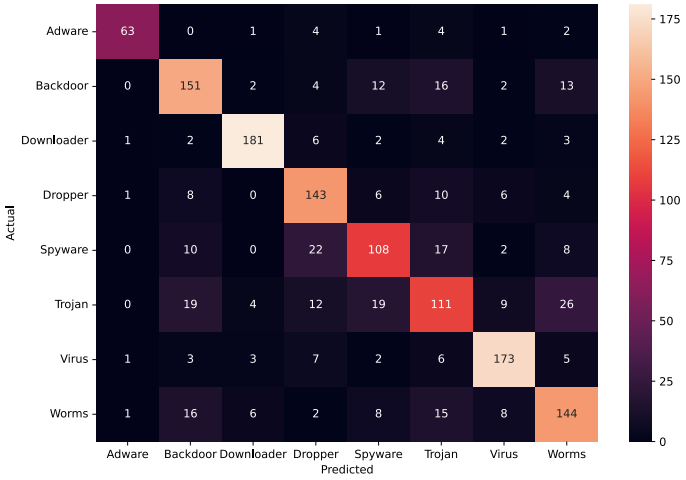

Fig. 6. CM SVC.

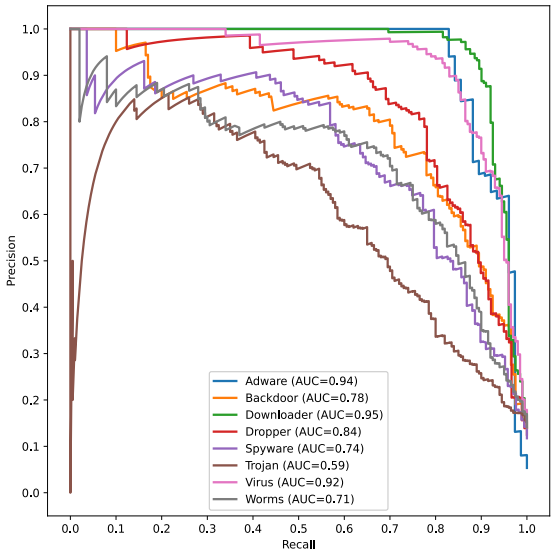

Fig. 7. Precision Recall of SVC.

TABLE I. PERFORMANCE: SVC VS. RF WITHOUT SELECTAPI FEATURE SELECTION

	Accuracy	Precision (Macro-Avg)	Recall (Macro-Avg)	F1-Score (Macro-Avg)	MCC
SVC	0.64	0.66	0.65	0.65	0.58
RF	**0.65**	**0.67**	**0.65**	**0.66**	**0.59**

TABLE II. PERFORMANCE: SVC VS. RF WITH SELECTAPI FEATURE SELECTION

	Accuracy	Precision (Macro-Avg)	Recall (Macro-Avg)	F1-Score (Macro-Avg)	MCC
SVC	**0.76**	**0.77**	**0.76**	**0.76**	**0.74**
RF	0.73	0.75	0.73	0.74	0.71

APIs for finalized API sequences of SelectAPI. The API embeddings are used to supply the expected weight matrix to train the SVC model with parameters (probability: 'True', kernel: 'RBF', gamma: 'auto', C: '10', and maximum iteration:

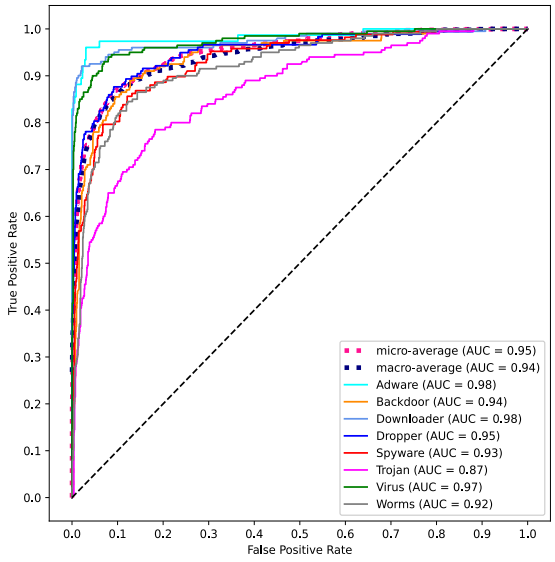

Fig. 8. ROC AUC of SVC.

'1000') and RF model with parameters (number of estimators: '100', criterion: 'entropy', bootstrap: 'True', and maximum depth: '150').

TABLE III. SVC USING SELECTAPI AGAINST OTHERS

	Accuracy	Precision (Macro-Avg)	Recall (Macro-Avg)	F1-score (Macro-Avg)
LSTM [4] (Single-Layer)	–	0.50	0.47	0.47
GRU [17] (Case2)	0.55	0.56	0.59	0.57
RTF Model [18]	0.60	–	–	0.61
SVC (SelectAPI)	**0.76**	**0.77**	**0.76**	**0.76**

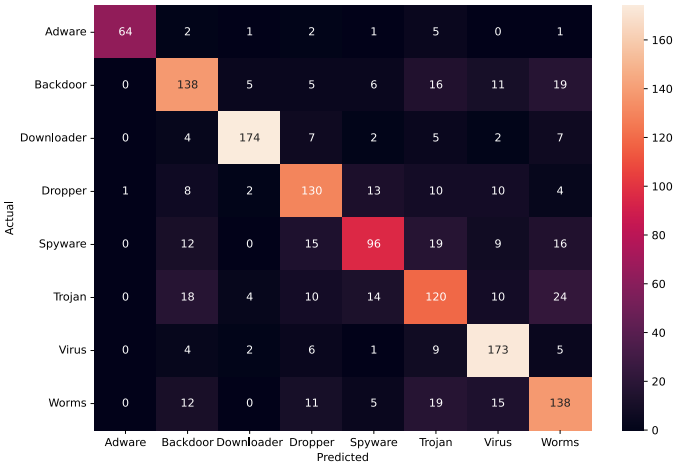

Fig. 9. CM RandomForest.

During experimentation, performance metrics for both the model's RF and SVC were recorded to have a conclusion about the significance of the suggested feature selection technique. Table I represents the accuracy, macro average precision, recall, and F1-score of the model's RF and SVC without

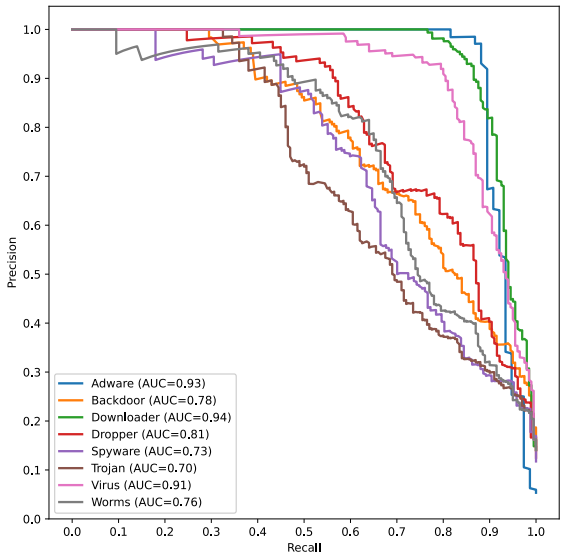

Fig. 10. Precision recall of RandomForest.

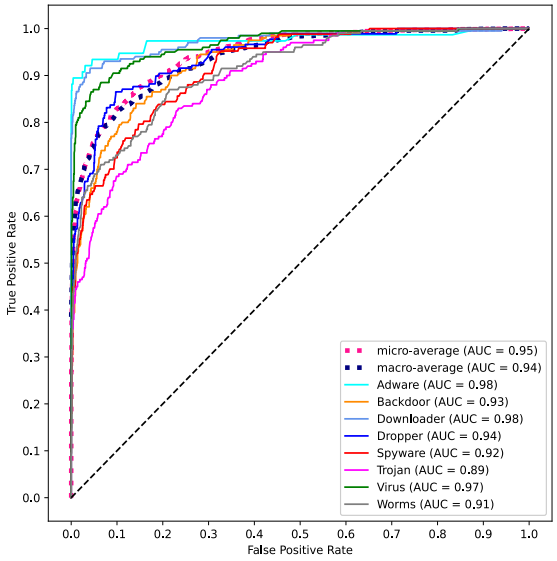

Fig. 11. ROC AUC of RandomForest.

using the feature selection SelectAPI. Furthermore, Table II highlights the performance metrics of both models with the feature selection SelectAPI. The SVC model's performance can be visually studied, referring to the following figures. The confusion matrix is in Fig. 6, the precision-recall curve is plotted in Fig. 7, and the ROC-AUC curve is plotted in Fig. 8 . The RF model's performance can be visually studied using the following figures. The confusion matrix is in Fig. 9, the precision-recall curve is plotted in Fig. 10, and the ROC-AUC curve is plotted in Fig. 11 . A comparison between performance scores mentioned in Table I and Table II reveals the significance of the feature selection technique SelectAPI. The Support Vector Classifier (SVC) obtained a detection accuracy of 0.76, according to the data, with overall average precision, recall, and F1-score values of 0.77, 0.76, and 0.76, respectively. The Random Forest (RF) model, in contrast, had overall average precision, recall, and F1-score values of 0.75, 0.73, and 0.74, respectively, and a detection accuracy of 0.73. As shown in Table II, the support value of 1422 is considered to evaluate all performance metrics. The Matthews correlation coefficient (MCC) score is also calculated to support the statistical significance of both models. All of the data clearly shows that SVC has performed better than RF in terms of classification abilities. This finding indicates a significant improvement over previous research that indicated a maximum macro average 'F1' score of 0.61 (see Table III).

V. Conclusion

This work has shown a method of selecting influential APIs from the collected API sequences for better malware classification. The significance of the suggested feature selection method, SelectAPI, is illustrated by applying it to the extremely unbalanced open benchmark variable-length API-sequence multiclass malware dataset MAL-API-2019. Selecting influential APIs from API call sequences improves the classification capability of malware to their classes even though the dataset is imbalanced class-wise and feature sequence length-wise. The SVC model demonstrated improved performance measures compared to RF as outlined in TABLE II, achieving an accuracy of 0.76 with a macro-avg. F1-score of 0.76 and a macro-avg. AUC-score of 0.94 across eight malware classes in the dataset using the feature section technique **SelectAPI**. These encouraging results highly support the effectiveness of the suggested feature section technique. Compared to previous studies by other researchers on the used dataset, as shown in Table III, which reported a maximum macro average F1 score of 0.61, the result obtained in this work shows a considerable performance improvement. Investigating other techniques to select influential APIs on such imbalanced dynamic API sequence-based malware datasets can give better classification capabilities to machine learning models.

References

[1] "Quick Heal Annual Report FY-2022-23, https://www.quickheal.co.in/documents/investors/quick-heal-annual-report-fy-2022-23.pdf," 2023.

[2] F. Cohen, "Computer viruses: Theory and experiments," *Computers & Security*, vol. 6, no. 1, pp. 22–35, 1987.

[3] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—a state of the art survey," *ACM Comput. Surv.*, vol. 52, no. 5, sep 2019.

[4] C. Ferhat Ozgur, Y. Ahmet Faruk, E. Ogerta, and A. Javed, "Deep learning based sequential model for malware analysis using windows exe api calls," *PeerJ Comput Science 6:e285*, 2020.

[5] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent pe-malware detection system based on association mining," *Journal in Computer Virology*, vol. 4, pp. 323–334, 11 2008.

[6] J. Geng, J. Wang, Z. Fang, Y. Zhou, D. Wu, and W. Ge, "A survey of strategy-driven evasion methods for pe malware: Transformation, concealment, and attack," *Computers & Security*, vol. 137, p. 103595, 2024.

[7] P. Louthánová, M. Kozák, M. Jureček, M. Stamp, and F. Di Troia, "A comparison of adversarial malware generators," *Journal of Computer Virology and Hacking Techniques*, vol. 20, no. 4, p. 623–639, 2024.

[8] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," 08 2013, pp. 1357–1365.

[9] Y. Ding, X. Yuan, K. Tang, X. Xiao, and Y. Zhang, "Malware detection based on objective-oriented association mining," *Computers & Security*, vol. 39, p. 315–324, 11 2013.

[10] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, 2017.

[11] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, "Maldae: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Computers and Security*, vol. 83, pp. 208–233, 2019.

[12] B. Panda and S. N. Tripathy, "Detection of anomalous in-memory process based on dll sequence," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 10, 2020.

[13] J. Mathew and M. Kumara, *API Call Based Malware Detection Approach Using Recurrent Neural Network—LSTM*, 01 2020, pp. 87–99.

[14] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Generation Computer Systems*, vol. 55, pp. 376–390, 2016.

[15] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers and Security*, vol. 81, pp. 123–147, 2019.

[16] B. Panda, S. S. Bisoyi, and S. Panigrahy, "An ensemble approach for imbalanced multiclass malware classification using 1d-cnn," *PeerJ Computer Science*, vol. 9:e1677, 2023.

[17] C. Li and J. Zheng, "Api call-based malware classification using recurrent neural networks," *Journal of Cyber Security and Mobility*, vol. 10, no. 3, pp. 617–640, May 2021.

[18] F. Demirkiran, A. Cayir, U. Unal, and H. Dag, "An ensemble of pre-trained transformer models for imbalanced multiclass malware classification," *Computers and Security*, vol. 121, p. 102846, 2022.

[19] A. Galli, V. La Gatta, V. Moscato, M. Postiglione, and G. Sperlì, "Explainability in ai-based behavioral malware detection systems," *Computers & Security*, vol. 141, p. 103842, 2024.

[20] L. Qian and L. Cong, "Channel features and api frequency-based transformer model for malware identification," *Sensors*, vol. 24, no. 2, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/2/580

[21] A. N. Irfan, S. Chuprat, M. N. Mahrin, and A. Ariffin, "A malware analysis approach for identifying threat actor correlation using similarity comparison techniques," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 12, 2024. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2024.0151258

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.