Malware Classification Using Low-Level Characteristics

Tuan Van Dao*, Hiroshi Sato, Masao Kubo, and Yasuhiro Nakamura

***Abstract*—Malware is growing at breakneck speed and has become a global problem. Malware detection has reached a high accuracy level of nearly 100%; however, malware classification is still challenging. Distinguishing and classifying different types of malware from each other is essential to better understanding how they can infect computers and devices, their threat level, and how to protect against them. Traditional malware classification works based on signature and behavior approaches. This approach is fragile in address with polymorphic and metamorphic malware. Moreover, because of the rapid development of several automatic malware creation tools, these methods cannot catch up to the speed of malware generation. Machine learning has handled most of today's problems with models ranging from simple to complex. Current studies focus on high-level characteristics of malware, which require high computational costs to detect and classify malware via complex neural network architectures, but the performance is still not groundbreaking. On the contrary, low-level characteristics still have much potential but are still not fully exploited. This study takes the advance of ensembling two low-level characteristic sets, including registers and opcodes, and selecting the appropriate features through the selection feature algorithm to increase performance and reduce computational costs. Proposed method outperformed previous works on two different malware data. This paper shows that extraction and selection features are no less critical than it is for architecture development.**

***Index Terms*—Malware classification, opcode, register, machine learning**

I. Introduction

Malware has seen exponential growth in recent years, with Kaspersky's detection system discovering an average of 380,000 new malware per day in 2021, an increase of 20,000 variants compared to 2020 [1]. According to the 2021 SonicWall report, 268,362 pieces of detected malware had never been seen before 2020 [2].

The main reason is that new malware variants can be easily created by accessing free, open-source malware creation tools such as TheFatRat [3], and Arbitrium-RAT [4]. Moreover, malware authors can easily create new malware variants by parsing and modifying existing malware formats [5]. Furthermore, malware authors use polymorphism in their malicious components to evade detection. In other words, malware that belongs to the same malware family and has the same malicious behavior can always be modified or obfuscated in various ways [6, 7]. As a result, traditional signature-based methods cannot keep up with this malware.

Analysts must understand the characteristics and behavior of malware to classify it into the correct family in advance. There are two basic methods commonly used to find characteristics of malicious code: static and dynamic analysis. The static analysis explores malicious code without executing it but requires advanced analysis to read assemblies and methods to unpack very complex, encrypted malicious code before debugging. Furthermore, malware authors use sophisticated programming, packing, and obfuscation tools to generate malware, making analysis more difficult. As a result, this approach is often time-consuming and labor-intensive.

The dynamic analysis captures and logs the behavior of malicious code through a virtual environment. This method performs better than static analysis because it correctly reveals process creation and registry operations, includes actual values of memory, variables, and registers, and often does not take as long to analyze as a static approach. However, malicious code that checks files, registry keys, and processes to detect virtual environments and immediately stops all activity as soon as it detects that the victim is not the target is becoming more common. In addition, some malware requires large amounts of resources to execute.

Security vendors and researchers have commonly employed machine learning and neural network approaches in recent years to address the limitations of static and dynamic analysis. Researchers can use machine learning-based methods to classify malware based on static characteristics.

In machine learning, input data is crucial in determining system performance. Some researchers have focused on high-level characteristics such as Application Programming Interface (API) calls, API arguments, instruction strings, and string information while missing research that exploits low-level characteristics such as opcodes and registers. Many studies have made progress in classification using natural language processing for high-level characteristics. However, this method cannot be applied to malware that uses multiple obfuscation methods and encryption; because the data obtained is noisy and the sequence is disorganized. N-grams, which create machine learning features from sequences of opcodes, are commonly used as low-level features. Value Set Analysis (VSA) is a typical method using registers and has proven to be effective in detecting metamorphic malware with an accuracy of up to 100% detection rate. Note that even if the malware modifies the sequence or adds noise, hidden or not, the detrimental features of the original code are preserved [8]. Also, the correlation between registers does not change when the malware applies register reassignment techniques. Thus, each malware family has a constant relationship between the opcodes and the registers.

This study focus on low-level features. This is because using low-level features allows us to recognize minor differences among families and requires less processing time

Manuscript received December 22, 2022; revised January 29, 2023; accepted March 28 , 2023 .

T. V. Dao, H. Sato, M. Kubo, and Y. Nakamura are with the Graduate School of Science and Engineering, National Defense Academy of Japan, Japan. E-mail: hsato@nda.ac.jp (H.S.), masaok@nda.ac.jp (M.K.), yas@nda.ac.jp (Y.N.).

*Correspondence: ed21006@nda.ac.jp (T.V.D.)

DOI: 10.7763/IJCTE.2023.V15.1339

than high-level features. This paper extracts opcodes and registers from a typical ASM file, which is a program written in the low level programming language known as assembly language, obtained by decompiling malicious code. This study uses a sequential feature selection algorithm to obtain critical feature families with solid correlation, further improving the classification results.

Instead of using each feature individually, this study has tried combining them and using a selection algorithm to choose the best combination. As a result, standard classifiers can produce high classification results, not inferior to models with complex structures, and use high-level characteristics.

II. Related Work

Yeboah et al. [9] employed ensemble features generated from opcode sequences of different n-gram sizes. The authors applied grid search to a predefined set of weights to find the optimal weights for the ensemble feature set. As a result, they achieved the best detection accuracy of 98.1% using Random Forest on a balanced dataset of 2,000 samples.

Rad et al. [10] propose and validate a classification method based on opcodes, which are statistical features of malware. While training and evaluation using Random Forest provide high performance, there is a possibility of overfitting due to a lack of data.

In preliminary experiments, *Li et al.* [11] proved that focusing on a single register alone does not yield good detection results. The authors use registers in a particular order: EAX, EBX, ECX, EDX, EBP, ESP, ESI, EDI, and consist of a simple Convolutional Neural Network (CNN) layer and three basic Long short-term memory (LSTM) layers. They achieve better performance than opcode sequences for all detection methods, including Accuracy, Precision, Recall, and F-score. However, the authors state that the highest detection accuracy is 0.796 when using machine learning algorithms. It is difficult for simple machine learning algorithms to detect malicious behavior because registers are microarchitecture-level features and cannot directly reflect information before being hidden in the dimensional space. The author states that it is difficult for simple machine learning algorithms to detect malicious acts. The paper also does not explain why the experiments were conducted in the fixed order described above.

Ghiasi et al. [12] applied the idea of VSA to track the distribution and change of register values through an executable file. Experiments showed that the authors' proposed method successfully discriminated samples from diverse datasets with low false positives. Experimental results show that the authors' proposed method successfully discriminates samples from diverse datasets with low false positives and achieves an average accuracy of more than 95% in distinguishing malware from benign software. However, it is only applicable when malware binaries are executed and monitored in a controlled environment. The system is based on six DLLs that are important for malicious activities of malware. Still, it also has the disadvantage of being computationally expensive due to the relatively large number of registered groups.

Vu et al. [13] focused on detecting metamorphic malware on Portable Executable (PE) files. They extracted four different features: API calls, PE headers, and DLLs imported from malware executables, which are formatted by PE format structure. They used the Longest Common Subsequence (LCS) by using the LCS algorithm that is better than the predefined threshold, as well as the N-gram method. They achieved a high F1-score with MLP at 93.0% with three hidden layers while low performance with SVM by only 79.5%.

Nguyen et al. [14] concentrated on the Windows API call frequency, which is extracted via dynamic analysis technique. The authors applied LightGBM, a gradient boosting framework that uses tree based learning algorithms to improve the accuracy and speed of classifying ransomware. They achieved a high F1-score at 95.2% with eight different types of ransomware.

III. Sequential Feature Selection Algorithm

Sequential feature selection algorithms are a family of greedy search algorithms used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace ($k < d$). The sequential feature selection algorithm is applied to reduce the dimension of the input to an appropriate number of features. The purpose of feature selection is two-fold. The first is to improve computational efficiency, and the second is to reduce the model's generalization error by eliminating irrelevant features and noise.

Sequential feature selection algorithms use SFS – Sequential Forward Selection (variable increasing) and SBS – Sequential Backward Selection (variable decreasing), where features are removed or added one by one until a feature subset of the desired size k is reached, based on classifier performance. SFS – Sequence Forward Selection and SBS – the Sequential Backward Selection removes or adds features one by one until a subset is reached. Floating methods are also used in the Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS) algorithms, which are extensions of SFS and SBS. The Floating method has an additional step of excluding or including features once included (or excluded) so that more feature subset combinations can be sampled. In this study, all feature selection methods are tested, and the method with the highest accuracy is investigated.

IV. Proposed Method

The classification pipeline is a multi-step process, as shown in Figure 1. First, load all executable files in the datasets divided into malware families to disassembler tool by Command Line Interface (CLI), produce ASM with the corresponding name and extension .asm. Note that a disassembler is a software tool that reads executable code and produces a human-readable representation of the machine instructions. Security researchers often use disassemblers to examine malware code and identify the methods attackers use to exploit system vulnerabilities. Then, 27 features are extracted from the ASM file with 19 standard opcodes, such as `MOV`, `PUSH`, `CALL`, `POP`, and eight registers. After extracting the statistics, a sequential feature selection algorithm is used to select an appropriate subset (k features) from the set of all features (27 features) to improve the accuracy.

This study uses typical machine learning classifier algorithms: k-nearest neighbors algorithm (k-NN), Nearest

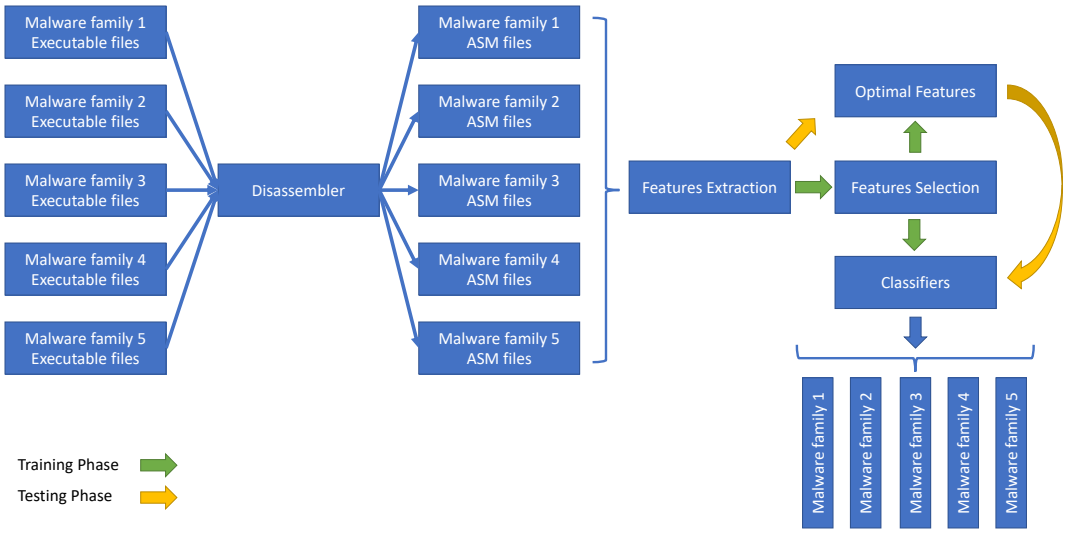


Fig. 1. Overview of proposed method.

TABLE I: Malware Dataset [13]

Malware Family	Number of sample
Locker	300
Mediyes	1,450
Winwebsec	4,400
Zbot	2,100
Zeroaccess	690
Total	8,940

TABLE II: Ransomware Dataset [14]

Ransomware Family	Number of sample
Reveton	522
TeslaCrypt	167
Win32:ransom	204
Win32:Cryptor	123
Win32:Crypt	146
LockScreen	123
WannaCry	491
Win32:FileCoder	27
Total	1,803

Centroid (NC), Support Vector Machines (SVM), Naive Bayes (NB), and Random Forest (RF). To evaluate proposed method, this paper uses 10-fold Cross-Validation. One of the ten subsamples is taken as validation data, and the remaining nine subsamples are used as training data. This process is repeated ten times, using each of the ten subsamples as validation data, and the average of the ten results is the quality of the method.

V. Experiments and Results

A. Dataset

In order to evaluate proposed approach, this study utilizes two datasets. The first is provided by [15] with five malware families and a total sample size of 8,940. All the malware files have been collected from VirusShare[1] and Malicia Project [16]. The number of samples for each malware family is shown in Table I. The second one belongs to [14] with eight malware families and a total sample size of 1,803 (Table II). The number of samples from the some families accounts for nearly half of the total, indicating that both datasets are unbalanced. Therefore, in addition to evaluating the performance by accuracy, this study also evaluates the model by F-score.

B. Evaluation Metrics

To evaluate the performance of the proposed CNN-based model, four standard performance metrics that are widely used in the existing research community were applied: accuracy, precision, recall, and F1-score. The four indicators are explained with the aid of the four parameters in Table III, where the class being evaluated is positive, and the remaining classes are negative.

Accuracy is defined as the ratio between the number of correctly classified samples to the total in the test dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision is defined as the ratios of true positive among the samples classified as positive.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall is the ratio of samples classified as positive among true positives.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F-score is used to evaluate the quality of the model.

$$F\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

[1] https://virusshare.com/

TABLE III: Performance Metric Parameters

Parameter	Description
True positive (TP)	The number of positive class samples that are correctly classified
True Negative (TN)	The negative class is correctly classified into the negative class
False Positive (FP)	The number of negative class samples misclassified into the possitive class
False Negative (FN)	The number of positive class samples misclassified into the negative class

C. Comparison of Feature Selection Method

A comparison of the four sequential feature selection algorithms is shown in Table IV and Table V. The best accuracy was obtained with the RF classifier as the SBFS of the feature selection methods on both datasets. The change in accuracy with the number of selected features is shown in Fig. 2 in terms of the best case.

According to Figure 2a, the best results were obtained when 13 features were selected out of 27, while 2b reached a peak with 12 features out of 27. In these cases, each dataset's selected opcode and register combinations are different. The former is 'mov,' 'push,''xor,' 'sub,' 'inc,' 'imul,' 'xchg,' 'shr,' 'cmp,' 'ror,' 'lea,' 'eax,' 'ebp.' The latter is 'push,' 'pop,' 'nop,' 'add,' 'imul,' 'shr,' 'call,' 'shl,' 'esi,' 'eax,' 'adi,' 'ebp.'

TABLE IV: Comparison of Sequential Feature Selection Method with Each Classifier on Dataset [13]

Feature Selection Method \ Algorithms	k-NN	NC	SVM	NB	RF
SFS	**98.88**	82.22	98.49	**81.94**	98.38
SBS	98.77	**90.78**	**98.60**	81.88	98.49
SFFS	98.49	82.06	98.38	81.88	98.88
SBFS	98.49	90.89	**98.60**	**81.94**	**98.99**

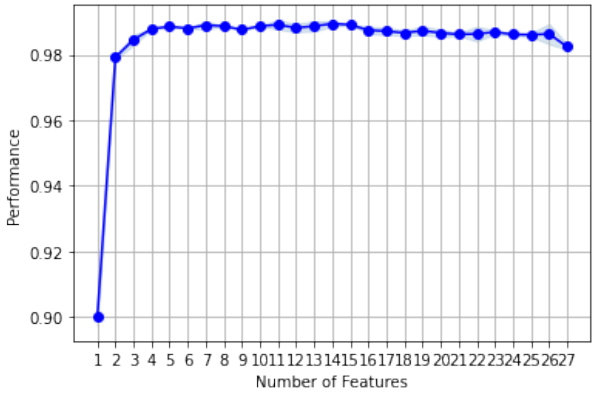
TABLE V: Comparison of Sequential Feature Selection Method with each classifier on Dataset [14]

Feature Selection Method \ Algorithms	k-NN	NC	SVM	NB	RF
SFS	**95.59**	81.69	**86.87**	71.66	96.23
SBS	94.94	80.55	85.32	**78.54**	95.37
SFFS	**95.59**	78.02	**86.87**	75.39	95.99
SBFS	94.65	**82.36**	86.59	77.46	**96.80**

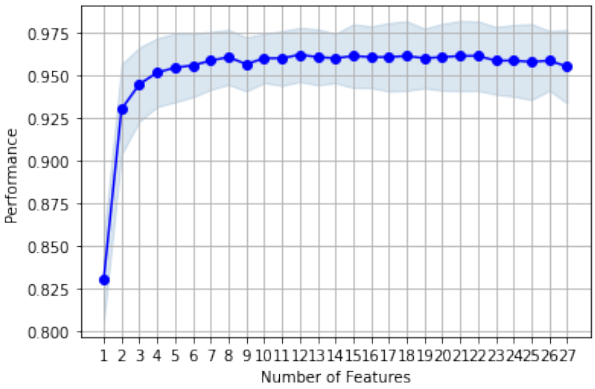
D. Experimental Results

The experiment compares accuracy and F-score when using opcodes and registers alone as input data to be processed by machine learning using both cases. Finally, when using feature selection.

According to Table VI and Table VII, High Accuracy and F-score were obtained for opcodes and registers with k-NN, SVM, and RF classifiers performing better than 85%. Although it was expected that the combination of opcode and register would yield better results, some experimental results showed that opcodes give better multiple features in cases of k-NN and SVM. In other words, this study found that simply incorporating non-selective malware features may not produce the desired results. Therefore, selecting features from a combination of low-level features



(a) SBFS with RF on malawre dataset [13]



(b) SBFS with RF on ransomware dataset [14]

Fig. 2. The best result of feature selection Method on both datasets.

allows machine learning to classify with better performance. Among the classifiers, the RF produced the best performance: on dataset [13], the Accuracy was improved by 0.05%, and the F-score was improved by 0.96% over combination without feature selection, the improvement is 0.05%, and 1.33%, respectively on dataset [14]. RF classifier is also effective in many malware classification tasks [9, 10].

Compared with *Vu et al.* [13] of study in Table VIII, the feature count has grown to over 3000 with just API calls. Although using numerous features, the results obtained are still not high. Compared to this study, with only 13 outstanding features selected from Opcodes and Registers, the F-score results are higher than the previous study by 4.5% on dataset [13].

Contrary to *Vu et al.* [13], *Nguyen et al.* [14] utilize only 286 different API calls but still get good performance. However, *Nguyen et al.* need to preprocess through the dynamic environment to get the malware's signature transformation. Proposed method does not need a dynamic environment to run malicious code and collect information as the authors

TABLE VI: Accuracy of Each Malware Classifier on Each Dataset

Datasets	Algorithms	Accuracy(%)			
		Opcodes	Registers	Opcodes and Registers	Opcodes and Registers (Feature Selection)
Malware dataset [13]	k-NN	98.77	98.21	98.71	**98.88**
	Nearest Centroid	80.55	70.88	81.16	**90.78**
	SVM	98.55	95.19	98.38	**98.60**
	Naive Bayes	80.82	70.58	80.87	**81.94**
	Random Forest	98.77	98.10	98.94	**98.99**
Ransomware dataset [14]	k-NN	95.79	95.35	95.46	**95.59**
	Nearest Centroid	77.89	73.19	78.82	**82.36**
	SVM	75.93	80.37	86.21	**86.87**
	Naive Bayes	74.68	70.19	75.33	**78.54**
	Random Forest	96.12	96.32	96.57	**96.80**

TABLE VII: F-score of Each Malware Classifier on Each Dataset

Datasets	Algorithms	F-score(%)			
		Opcodes	Registers	Opcodes and Registers	Opcodes and Registers (Feature Selection)
Malware dataset [13]	k-NN	96.10	94.54	96.01	**96.68**
	Nearest Centroid	66.43	45.44	67.01	**67.85**
	SVM	94.86	84.61	94.40	**95.30**
	Naive Bayes	58.15	35.50	58.25	**60.23**
	Random Forest	96.36	94.62	96.58	**97.54**
Ransomware dataset [14]	k-NN	87.32	85.07	85.21	**85.93**
	Nearest Centroid	58.73	55.69	55.77	**60.23**
	SVM	89.32	87.90	88.04	**88.44**
	Naive Bayes	49.16	42.21	50.52	**51.79**
	Random Forest	94.10	93.29	94.93	**96.26**

TABLE VIII: Comparison of Results with Existing Study

Dataset	Studies	Precision	Recall	F-score
Malware dataset	Vu et al. [13]	92.6	93.7	93.0
	Proposed method	**97.8**	**97.3**	**97.5**
Ransomware dataset	Nguyen et al. [14]	95.0	95.6	95.2
	Proposed method	**96.8**	**95.8**	**96.3**

do; the new malware can detect the control environment and stop working or intentionally run normal processes. Usually until executed in the natural environment. As a result, *Nguyen et al.*'s method becomes ineffective against such malware. According to Table VIII, proposed method achieved a higher F-score than the authors by 1.3% on the dataset [14].

VI. Conclusions and Future Work

Many security vendors have utilized the application of AI in antivirus software due to its high performance and processing capacity compared to traditional methods that require human analysis. With traditional methods' existing tools, the malicious code's properties are extracted and utilized as input for machine learning. Current research tends to build complex models as well as use many high-level features. However, the expected results have yet to be achieved. Besides, taking advantage of low-level characteristics (opcodes and registers) is still a problem in choosing appropriate attributes to increase performance for machine learning. It was found that increasing the number of features by simply combining opcodes and registers does not necessarily improve accuracy. This study applied a sequential feature selection algorithm to obtain a better combination of features, resulting in improved classification performance. In the experiments, proposed method obtained an accuracy of 98.99% and an F-score of 97.54% on a multitype of malware dataset, 96.80% and 96.26%, respectively on a ransomware dataset which is comparable to existing studies with high-level features. Furthermore, considering the optimal combination of features obtained from the sequential feature selection algorithm may be helpful as a hint for advanced malware analysis.

Using many different attributes does not always necessarily bring the expected results. This study also shows the

potential for high performance with the appropriate features. Proposed method overperformed the previous study in all evaluation metrics in the experimental result with only a much smaller number of features. As a result, the proposed approach can be easily extracted and applied to actual malware classification.

In this study, high classification accuracy was achieved with a simple method. However, this method requires a large amount of data, and ordinary debugging tools may not correctly disassemble malicious code created by targeted attacks or zero-day malware. Noise in the data, such as when malware authors modify the code to make it more complex or use various encodings, can significantly affect statistical methods. In feature work, other methods will be collaborated to reduce the effect of noise as much as possible. Besides, recent studies also indicate that adversarial attacks can trick machine learning models by providing deceptive input. The upcoming work will measure the robustness of the proposed method with these attacks.

Conflict of Interest

The authors declare no conflicts of interest.

Author Contributions

Conceptualization, T.V.D. and H.S.; methodology, T.V.D.; validation, H.S., M.K. and Y.N.; writing-original draft preparation, T.V.D.; writing-review and editing, H.S., M.K. and Y.N. All authors have read and agreed to the published version of the manuscript.

References

[1] Kaspersky Lab, "Digital life deserves complete protection: Kaspersky announces early access to new and reimagined consumer product portfolio", Press release, Aug. 04, 2022, https://www.kaspersky.com/about/press-releases/2022_digital-life-deserves-complete-protection-kaspersky-announces-early-\\access-to-new-and-reimagined-consumer-product-portfolio

[2] SonicWall, "2021 Cyber Threat Report ", https://www.sonicwall.com/resources/white-papers/2021-sonicwall-cyber-threat-report

[3] https://github.com/screetsec/TheFatRat

[4] https://github.com/ToR-0/Arbitrium-RAT

[5] LIEF project, "Library to Instrument Executable Formats", https://github.com/lief-project/LIEF

[6] D. Gibert, C. Mateu, J. Planes and R. Vicens, "Classification of Malware by Using Structural Entropy on Convolutional Neural Networks," The Thirtieth AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18), in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.32, no.1, pp.7759–7764, 2018.

[7] J. Singh and J. Singh, "Challenges of Malware Analysis: Obfuscation Techniques," in *International Journal of Information Security Science*, vol.7, no. 3, pp. 100–110, 2019.

[8] F. Leder, B. Steinbock, and P. Martini, "Classification and Detection of Metamorphic Malware using Value Set Analysis," in *4th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 39–46, Otc. 2009.

[9] P. N. Yeboah, S. K. Amuquandoh, and H. Balle, "Malware Detection Using Ensemble N-gram Opcode Sequences," in *International Journal of Interactive Mobile Technologies (iJIM)*, Vol. 15, No. 24, pp. 19–31, 2021.

[10] B. B. Rad, M. Masrom, S. Ibrahim, and S. Ibrahim, "Morphed Virus Family Classification Based on opcodes Statistical Feature Using Random Forest," in *International Conference on Informatics Engineering & Information Science*, pp. 123–131, Nov. 2011.

[11] F. Li, C. Yan, and Z. Zhu, "A Deep Malware Detection Method Based on General-Purpose registers Features", in *19th International Conference, Faro, Portugal*, pp. 221–235, Jun. 2019.

[12] M. Ghiasi, A. Sami, and Z. Salehi, "Dynamic malware detection using registers values set analysis," in *9th International ISC Conference on Information Security and Cryptology*, pp. 54–59, Sep. 2012.

[13] T. N. Vu, T. T. Nguyen, H. Phan Trung, T. Do Duy, K. H. Van, and T. D. Le, "Metamorphic malware detection by PE analysis with the longest common sequence," In *International Conference on Future Data and Security Engineering*, vol. 10646, pp. 262–272, Nov. 2017.

[14] D. T. Nguyen and S. Lee, "LightGBM-based Ransomware Detection using API Call Sequences," In *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, 2021.

[15] A. P. Tuan, A. T. H. Phuong, N. Vu. Thanh, T. N. Van, "Malware Detection PE-Based Analysis Using Deep Learning Algorithm Dataset," figshare, 2018, https://doi.org/10.6084/m9.figshare.6635642.v1.

[16] A. Nappa, M. Z. Rafique, and J. Caballero, "The MALICIA dataset: identification and analysis of drive-by download operations," in *International Journal of Information Security*, vol. 14, no. 1, pp. 15–33, 2015.