An Efficient Framework for Multi-Class Malware Classification in Cloud Environments

DOI: https://doi.org/10.54654/isj.v1i24.1092

Pham Sy Nguyen, Pham Ngoc Van, Hoang Viet Long, Pham Duy Trung*

***Abstract*— Malware classification in cloud environments remains a critical challenge due to the increasing complexity and volume of cyber threats. This paper proposes CMC (Cloud-based Malware Classification), a novel framework that enhances multi-class malware classification efficiency through the integration of feature selection, dimensionality reduction, and imbalanced data handling techniques. The CMC framework aims to improve classification accuracy and computational efficiency by optimizing feature representation and addressing class imbalance, which are common issues in real-world malware datasets. To evaluate its effectiveness, we apply the proposed model to two public benchmark datasets: CMD_2024 and CIC-MalMem-2022. Experimental results demonstrate that CMC outperforms existing approaches in terms of classification accuracy, F1-score, and computational efficiency, proving its potential for real-world deployment in cloud-based security solutions. These findings highlight the importance of intelligent data preprocessing and feature optimization in enhancing malware classification on cloud platforms.**

***Tóm tắt*— Phân loại phần mềm độc hại trên môi trường đám mây vẫn là một thách thức quan trọng do tính phức tạp và khối lượng ngày càng tăng của các mối đe dọa mạng. Bài báo này đề xuất CMC (Phân loại phần mềm độc hại trên đám mây), một khuôn khổ mới giúp tăng cường hiệu quả phân loại phần mềm độc hại đa lớp thông qua việc tích hợp lựa chọn tính năng, giảm chiều và các kỹ thuật xử lý dữ liệu mất cân bằng. Khuôn khổ CMC nhằm mục đích cải thiện độ chính xác của phân loại và hiệu quả tính toán bằng cách tối ưu hóa biểu diễn tính năng và giải quyết tình trạng mất cân bằng lớp, đây là những vấn đề phổ biến trong các tập dữ liệu phần mềm độc hại trong thế giới thực. Để đánh giá hiệu quả của khuôn khổ đề xuất, chúng tôi áp dụng CMC cho hai tập dữ liệu chuẩn công khai: CMD_2024 và CIC-MalMem-2022. Kết quả thử nghiệm chứng minh rằng CMC vượt trội hơn các phương pháp hiện có về độ chính xác phân loại, điểm F1 và hiệu quả tính toán, chứng minh tiềm năng triển khai trong thế giới thực của nó trong các giải pháp bảo mật dựa trên đám mây. Những phát hiện này làm nổi bật tầm quan trọng của việc xử lý dữ liệu thông minh và tối ưu hóa tính năng trong việc tăng cường phân loại phần mềm độc hại trên nền tảng đám mây.**

***Keywords*— Malware Classification; Random Undersampling; Low Variance Filtering and Scaling; Cloud-based Malware Detection.**

***Từ khóa*— Phân loại phần mềm độc hại; Lấy mẫu ngẫu nhiên; Lọc phương sai thấp và chuẩn hoá; Phát hiện phần mềm độc hại trên nền tảng đám mây.**

I. INTRODUCTION

Malware detection and classification have become increasingly critical in today's cloud-centric environment, where cyber threats rapidly evolve and target diverse infrastructures [1-3]. The proliferation of malware, including viruses, trojans, worms, and ransomware, poses severe risks to both individual users and organizations operating on cloud platforms. Traditional malware detection approaches, such as signature-based, heuristic-based, and behavior-based techniques, have been enhanced with Machine Learning (ML) and Deep Learning (DL) models to improve detection accuracy and scalability [4-6]. However, the effectiveness of these models is often hindered by several challenges, including imbalanced datasets and high-dimensional feature spaces, which significantly impact classification performance.

Recently, various publicly available malware datasets have been introduced to facilitate research in this field, including CIC-MalMem-2022 [7] and CMD_2024 [8].

This manuscript was received on April 25, 2025. It was reviewed on May 13, 2025, revised on x/dd/yy and accepted on June 2, 2025.

* Corresponding author.

These datasets provide valuable insights into malware behavior but also exhibit notable limitations that affect classification performance. The primary challenges associated with malware classification on cloud environments are as follows:

- **Imbalanced Datasets**: Malware datasets are often dominated by benign samples, leading to a lack of representative malicious samples. This imbalance causes classifiers to favor the majority class, reducing the detection accuracy of minority malware families.
- **High-Dimensional Data**: Malware datasets typically contain a large number of features, many of which are redundant or non-informative. Retaining irrelevant features can lead to overfitting and increased computational costs, making efficient dimensionality reduction a necessity.
- **Real-Time Classification**: Cloud-based security solutions demand fast and accurate classification models to handle large-scale, real-time data streams effectively. Computationally expensive models struggle to meet these requirements.

To address these challenges, we propose the Cloud-based Malware Classification (CMC) framework, which integrates feature selection, dimensionality reduction, imbalanced data handling, and gradient boosting-based classification to enhance malware detection in cloud environments. The key components of CMC are as follows:

- Variance-Based Dimensionality Reduction (VDR): CMC eliminates low-variance features that contribute little to classification performance. By removing features with negligible variance across samples, the framework reduces model complexity while preserving relevant information.
- Synthetic Minority Oversampling Technique (SMOTE) combined with Random Undersampling: To address dataset imbalance, CMC applies a combination of SMOTE [9] and Random Undersampling. SMOTE generates synthetic samples for the minority malware classes by interpolating between existing instances, enhancing model generalization and improving classification performance for rare malware families. Simultaneously, Random Undersampling is employed to reduce the number of samples from the majority class, thereby balancing the class distribution and improving the model's accuracy and stability.
- Soft Voting-Based Classification with DNN and LGBM: CMC employs a Soft Voting approach by combining Deep Neural Networks (DNN) and LightGBM (LGBM) for malware classification. This approach leverages the strengths of both models, where DNN is used for learning complex patterns from the data, and LGBM provides efficient handling of high-dimensional, imbalanced datasets. The Soft Voting mechanism aggregates the probability outputs from both models to make the final classification decision, ensuring improved robustness and accuracy. This ensemble method enhances classification performance by balancing the strengths of DNN's ability to model non-linear relationships and LGBM's efficiency in dealing with large datasets, offering better scalability and faster training compared to traditional classifiers.

The main contributions of our work are:

1. Efficient Feature Selection and Dimensionality Reduction: The application of Low Variance Filtering and Scaling reduces computational overhead by eliminating features with low variance, while preserving the essential features needed for accurate classification.
2. Balanced Malware Classification: The combination of SMOTE and Random Undersampling effectively addresses dataset imbalance, improving detection rates for underrepresented malware families.
3. Optimized Soft Voting Model with DNN and LGBM: The CMC framework utilizes a Soft Voting ensemble model combining DNN and LGBM. This hybrid approach ensures high accuracy and computational efficiency, making it ideal for real-time malware classification in cloud-based environments. By aggregating the predictions from both DNN and LGBM through Soft Voting, the framework leverages DNN's ability to capture complex patterns and LGBM's efficiency in handling large-scale, imbalanced datasets. The result is a robust and scalable classification model

capable of delivering high performance in dynamic, real-time scenarios.
4. Extensive Evaluation on Public Datasets: CMC is thoroughly tested on CMD_2024 and CIC-MalMem-2022 datasets, showcasing superior classification performance compared to baseline models.

The remainder of this paper is structured as follows. Section II provides the background, discussing the fundamental concepts and challenges in malware detection and classification. Section III reviews related work in the field, highlighting existing malware classification approaches and their limitations. Section IV details the proposed CMC framework, including techniques for feature selection, data balancing, and classification. Section V presents experimental results and an evaluation of the framework's performance on CMD_2024 and CIC-MalMem-2022 datasets. The paper concludes with key findings and future research directions.

II. Background

A. *Cloud-based Malware Detection and Classification*

Malware detection and classification play a crucial role in modern cloud security. The emergence of polymorphic and metamorphic malware, which dynamically alter their structure and behavior, reduces the effectiveness of static analysis techniques [10]. Although dynamic analysis enhances detection accuracy, it requires substantial computational resources, making it less feasible for large-scale, cloud-based environments [11, 12].

Cloud environments, provide an ideal infrastructure to address these challenges. By leveraging the scalability, distributed processing power, and resource pooling offered by the cloud, it becomes feasible to handle the high demands of malware detection and classification in real-time. The cloud enables efficient monitoring and extraction of malware-related data from vast networks of cloud-based devices, servers, and virtual machines. In cloud-based environments, malware detection is enhanced by the ability to continuously monitor large volumes of data across multiple virtual machines and containers. These cloud resources can be distributed to track real-time system behavior, including network activity, file operations, system calls, and API interactions, all of which are indicative of malware behavior. The cloud infrastructure allows for the extraction of relevant features from these data streams, which can be processed and analyzed using advanced machine learning and deep learning models.

Another challenge in malware classification is the high dimensionality of malware datasets. These datasets often contain redundant or non-informative features, which increase computational complexity and degrade the performance of ML models. Feature selection and dimensionality reduction techniques, such as variance-based filtering, are crucial to mitigating these issues by retaining only the most informative features while discarding those with minimal variance.

Additionally, data imbalance remains a major concern, as benign samples typically outnumber malicious ones, leading to biased models with poor detection rates for minority malware families [13]. Traditional classifiers struggle with such imbalances, favoring the majority class and reducing the ability to detect rare malware variants. To address this, Synthetic Minority Oversampling Technique (SMOTE) has been widely adopted to generate synthetic samples for underrepresented classes, improving classification performance.

A key challenge in malware classification is ensuring model generalizability across different platforms and environments. Malware behavior varies across operating systems, cloud infrastructures, and attack vectors, limiting the adaptability of models trained on specific datasets. Addressing these issues requires scalable solutions that handle high-dimensional data, mitigate class imbalance, and efficiently detect evolving malware threats.

B. *ML/DL Approaches for Malware Detection*

ML and DL have been widely utilized in malware detection and classification tasks. Traditional ML techniques, including Support Vector Machines (SVMs), Random Forest (RF), and k-Nearest Neighbors (kNN), have demonstrated effectiveness in extracting patterns from malware datasets [14 - 16]. These methods typically rely on handcrafted features, making

them highly dependent on feature engineering. On the other hand, DL models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), can automatically learn complex and hierarchical representations from malware data, significantly improving detection accuracy [17], [18].

Recent studies have explored the application of deep learning architectures for malware classification. For example, CNN-based models have been successfully applied to classify malware binaries based on their image representations [19]. Meanwhile, RNNs and Long Short-Term Memory (LSTM) networks have been leveraged to detect sequential patterns in malware execution logs [20], [21]. Nguyen et al. [22] present a new approach that leverages deep graph learning for dynamic malware analysis. By representing program behavior as a graph, the authors use deep graph learning techniques to capture both structural and dynamic features of PE files. This allows for more robust detection of sophisticated malware that might evade traditional feature-based detection methods. The study demonstrates that deep graph learning significantly enhances the classification accuracy in dynamic analysis, providing a more effective solution for malware detection in real-world environments. Dinh et al. [23] focus on the importance of selecting dynamic features for malware classification. They propose a behavior-aware dynamic feature selection method that adapts to the evolving characteristics of malware. This approach enables the model to select the most relevant features from execution traces, improving the performance of malware detection systems. The study's results show that dynamic feature selection leads to better classification outcomes, especially when combined with behavior-aware models, thus advancing the field of dynamic malware detection.

Despite these advancements, several critical challenges remain, particularly in the context of cloud-based malware classification:

- **Limited Feature Optimization**: Most ML/DL models do not employ effective feature selection methods. Retaining irrelevant features leads to overfitting and excessive training costs.
- **Data Imbalance Not Properly Addressed**: While techniques like oversampling and undersampling have been used to balance datasets, they often fail to preserve real-world data distributions, reducing model effectiveness in detecting rare malware families.
- **High Computational Overhead**: Many existing models rely on deep learning architectures (e.g., CNNs, LSTMs) that demand substantial computational resources, making them impractical for real-time malware detection in cloud environments.

III. Related Work

The challenge of handling imbalanced datasets in malware classification has been extensively addressed in recent research, with various techniques proposed to mitigate the impact of class imbalance. In this section, we review the methodologies proposed by several researchers, including feature extraction methods, resampling strategies, and ensemble learning approaches.

Çayır et al. [24] (2021) proposed a Random CapsNet Forest Model for imbalanced malware classification, using Capsule Networks (CapsNet) to reduce model complexity and improve generalization. By combining CapsNet with Bootstrap Aggregating (Bagging), they enhanced detection of underrepresented malware families. However, they noted that careful hyperparameter tuning is necessary to avoid overfitting. Demirkıran et al. [25] (2022) focused on transformer-based models like BERT and CANINE for imbalanced multi-class malware classification. They introduced the Random Transformer Forest (RTF) ensemble model, which combines multiple transformer models using bagging. Their results showed that RTF outperformed traditional models, achieving an F1-Score of 0.6149, demonstrating the effectiveness of transformer-based ensembles in handling class imbalance. Similarly, Zhu et al. [13] (2023) introduced the MEFDroid framework, which used hybrid deep learning models like Sparse Autoencoders (SAE) and Deep Autoencoders (DAE) to extract features and applied ensemble learning to improve classification. While achieving high performance with an F1-score of 97.12%, they highlighted the

dependency on classifier choice and feature selection.

Xue et al. [26] (2024) proposed a hybrid framework combining oversampling, undersampling, and weighted majority voting classifiers (WMVC) to address multi-class anomaly detection in imbalanced malware datasets. They used SMOTE and Tomek Links to balance the data, achieving an F1-score of 97.12%. However, they noted challenges such as overfitting and the complexity of tuning ensemble classifiers. Latif et al. [27] (2024) proposed a hybrid model combining Vision Transformer (ViT) and 1D Convolutional Neural Network (1DCNN) for ransomware classification to address data imbalance. They used SMOTE to balance the dataset, achieving 98% accuracy while reducing false positives and negatives. However, oversampling introduced noise, which could impact real-world performance. Andelić et al. [28] (2025) focused on Android malware detection using graph-based models and ensemble learning. Their method, combining symbolic classifiers with multi-ensemble threshold techniques, achieved 99.99% accuracy and an F1-score of 82.15%. Despite these successes, precision and recall were limited, particularly for certain attacks with misclassifications. Putra et al. [29] (2025) introduced a graph-based botnet detection model using network flow analysis and adaptive weighting. The model used IP addresses as nodes and communication links as edges, with 16 weighting methods. Despite strong performance, the model struggled with detecting low-intensity attacks and faced precision issues in some cases.

These studies emphasize the importance of feature selection, resampling techniques like SMOTE, and ensemble learning in overcoming imbalanced datasets for malware detection. While approaches integrating deep learning and hybrid models have improved classification performance, challenges like overfitting, model complexity, and the need for further optimization remain. Motivated by these limitations, our CMC framework is designed to optimize multi-class malware classification by effectively addressing these issues, improving model robustness and performance in real-world scenarios.

IV. Proposed Framework: CMC

The proposed CMC framework (Figure 1) is designed to improve malware detection and classification, especially for imbalanced datasets, by combining advanced preprocessing techniques, data balancing strategies, dimensionality reduction, and ensemble learning. The framework is organized into three distinct phases, each targeting a specific challenge in the malware detection process.

A. Phase 1: Data Preprocessing and Feature Selection

In the first phase, the original dataset undergoes preprocessing to prepare it for further analysis. This includes applying Low Variance Filtering and Scaling, which helps to remove irrelevant features with minimal variance, thereby reducing the dimensionality of the dataset. The dataset is then split into two sets: 80% for training and 20% for testing. The feature selection process is achieved using Low Variance Filtering, which enhances the feature set by retaining the most relevant and informative features while discarding those with little variance. This ensures that the dataset is clean, well-scaled, and ready for the subsequent stages of data balancing and classification.

To further enhance model performance and prevent scale-induced bias, we apply Standardization (Z-score normalization) to ensure all features have a mean of zero and unit variance. This step improves model stability by making features comparable across different scales.

1. Low Variance Filtering

Variance measures the spread of values in a feature. Features with very low variance provide minimal useful information for classification. Therefore, we remove any feature f_i whose variance V_i is below a predefined threshold τ:

$$V_i = \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \bar{x}_i)^2 \tag{1}$$

where:

- n is the number of samples in the dataset.
- x_{ij} is the value of feature f_i in sample j.

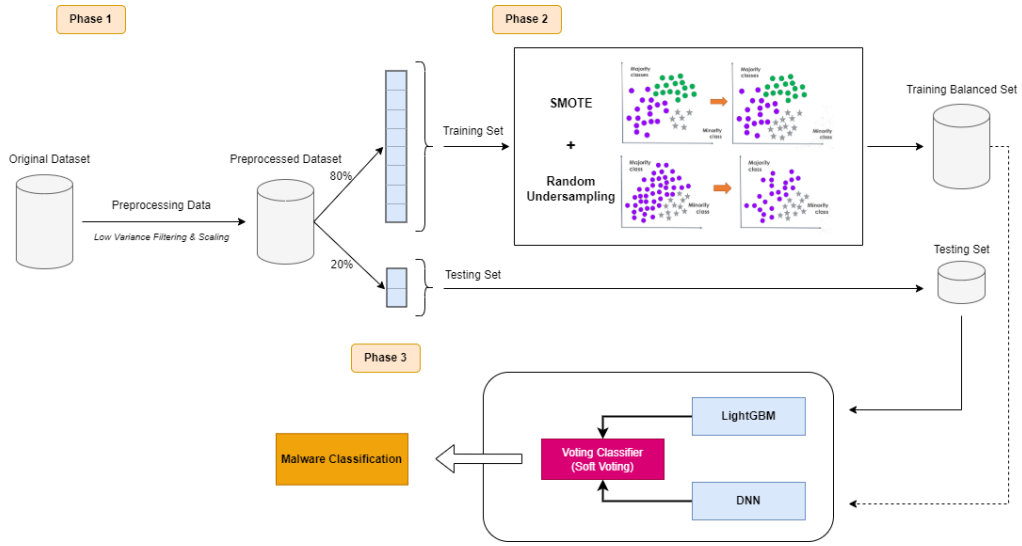


Figure 1. CMC Framework Detailed Architecture

- $\bar{x}_i$ is the mean of feature f_i, given by:

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \tag{2}$$

A feature f_i is removed if:

$$V_i < \tau \tag{3}$$

where τ is a small predefined threshold (e.g., 0.01).

2. Feature Standardization

After selecting relevant features, we apply Z-score normalization (also known as standardization) to ensure that all features have a mean of zero and a standard deviation of one. The transformation is defined as:

$$x'_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i} \tag{4}$$

where:

- μ_i is the mean of feature f_i, calculated using Equation 2.
- σ_i is the standard deviation of feature f_i, given by:

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \mu_i)^2} \tag{5}$$

This transformation ensures that features are on the same scale, preventing those with larger magnitudes from dominating the learning process.

The detailed procedure is outlined in Algorithm 1.

Algorithm 1 Data Preprocessing and Feature Selection

Input:
$\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ – Input with features F and labels Y.
τ – Variance threshold for feature selection.
r – Train/Test split ratio; default 0.2 (e.g., 80% training, 20% testing).

Output:
T – Training set.
V – Testing set.

$(X, Y) \leftarrow$ ExtractFeaturesAndLabels$(\mathcal{D})$ ▷ Separate features and labels
$X' \leftarrow X$ ▷ Initialize feature set
for $f_i \in X$ **do** ▷ Apply Low Variance Filtering
  $V_i \leftarrow$ ComputeVariance(f_i)
  if $V_i < \tau$ **then**
    $X' \leftarrow X' \setminus \{f_i\}$
  end if
end for
$X_s \leftarrow$ StandardScaler(X') ▷ Normalize features
$Y' \leftarrow$ LabelEncoder(Y) ▷ Label encoding
$(T, V) \leftarrow$ SplitTrainTest(X_s, Y', r) ▷ Train/Test split
return (T, V)

B. Phase 2: Data Balancing Using SMOTE and Random Undersampling

Phase 2 focuses on addressing class imbalance, a major challenge in malware detection. To achieve this, the dataset is balanced using a combination of SMOTE and Random Undersampling. SMOTE generates synthetic samples by interpolating between existing instances in the minority classes, ensuring adequate representation. Meanwhile, Random

Undersampling reduces the size of the majority class, preventing dominance over other categories.

This process produces a balanced training set, mitigating bias towards majority classes and enhancing classification performance across all categories. The detailed balancing procedure is outlined in Algorithm 2.

Algorithm 2 Data Balancing with Random Undersampling and SMOTE

Input:
$T = (X_T, Y_T)$ – Imbalanced training set.
$V = (X_V, Y_V)$ – Testing set (remains unchanged).
θ – Maximum allowed size for majority classes; default $\theta = 2000$.
ρ – Minimum target size for minority classes; default $\rho = 2000$.
Output:
$T' = (X_{T'}, Y_{T'})$ – Balanced training set.
$V = (X_V, Y_V)$ – Testing set (unchanged).
$(S_{\text{maj}}, S_{\text{min}}) \leftarrow$ GetClassDistribution(T) ▷ S_{maj} = majority classes, S_{min} = minority classes
$T' \leftarrow T$ ▷ Initialize T' as the original training set
if $S_{\text{maj}} \neq \emptyset$ **then** ▷ Reduce majority classes
    $\mathcal{M} \leftarrow \{c \mid |c| > \theta\}$ ▷ $\mathcal{M}$ = Set of classes exceeding θ
    $T' \leftarrow$ RandomUndersample$(T', \mathcal{M}, \theta)$ ▷ Apply undersampling to $\mathcal{M}$
end if
$(S'_{\text{maj}}, S'_{\text{min}}) \leftarrow$ GetClassDistribution(T') ▷ Recompute $S'_{\text{maj}}, S'_{\text{min}}$ after undersampling
if $S'_{\text{min}} \neq \emptyset$ **then** ▷ Increase minority classes
    $\mathcal{L} \leftarrow \{c \mid |c| < \rho\}$ ▷ $\mathcal{L}$ = Set of classes below ρ
    $T' \leftarrow$ SMOTE$(T', \mathcal{L}, \rho)$ ▷ Apply SMOTE to $\mathcal{L}$
end if
return T', V

C. Phase 3: Malware Classification Using Soft Voting with DNN and LGBM

In the final phase, the balanced dataset is classified using a Soft Voting ensemble combining Deep Neural Networks (DNN) and LightGBM (LGBM). This method leverages the complementary strengths of both models: DNN effectively captures complex, non-linear patterns in malware behavior, while LGBM handles high-dimensional, imbalanced data efficiently with fast training times.

The DNN model extracts hierarchical features through multiple layers, enabling the detection of intricate patterns. In parallel, LGBM—a gradient boosting framework—is optimized for large-scale learning, with adjustments to mitigate class imbalance and overfitting.

Soft Voting aggregates the prediction probabilities from both models and assigns the final label based on the weighted average. This ensemble enhances overall classification performance and robustness, making it suitable for real-time, large-scale malware detection tasks under imbalanced conditions.

V. Experiments And Evaluation

This section describes the experimental evaluation of the proposed CMF framework, focusing on its application to two malware datasets. The experiments assess the framework's effectiveness in handling imbalanced data and improving malware detection and classification accuracy. The datasets, experimental setup, and results are detailed below. The framework is evaluated through the three phases: data preprocessing and feature selection, data balancing, and malware classification using ensemble learning.

A. Datasets

To evaluate the CMF framework, we use two public malware datasets with differing characteristics in class distribution, feature types, and imbalance levels, providing a robust benchmark. Table 1 summarizes class distributions before and after balancing.

The balancing strategy from *Phase 2*—SMOTE combined with Random Undersampling—addresses class imbalance by synthetically augmenting minority classes and reducing majority samples. This preserves natural distribution while focusing on hard-to-classify regions, improving learning and reducing bias toward dominant classes.

As a result, the adjusted distributions (Table 1) show improved balance, boosting classification performance, particularly for rare malware types.

CMD_2024: This dataset contains 20,850 samples across eight malware families: Trojan, Virus, Worm, Ransomware, Adware, Miner, PUA, and Downloader. Features include both static (e.g., file size, entropy, byte sequences) and

dynamic attributes (e.g., system calls, API calls, memory usage). Low Variance Filtering is applied to retain only informative features, enhancing classifier performance.

As shown in Table 1, the CMD_2024 dataset exhibits extreme imbalance, with minority classes like Downloader (26) and PUA (39) significantly underrepresented compared to the dominant class (>5000 samples). In contrast, CIC-MalMem-2022 shows a more moderate imbalance, with malware classes ranging from 1100–1900 samples, and a dominant benign class (>13,000).

To address this, we applied dataset-specific balancing strategies. For CMD_2024, we set a uniform target of 5000 samples per class ($\theta = \rho = 5000$), enhancing minority representation and reducing majority dominance. For CIC-MalMem-2022, a lower threshold of 2000 ($\theta = \rho = 2000$) was chosen to reflect its less severe imbalance. These thresholds were determined empirically to maintain class diversity while improving balance.

Although SMOTE and Random Undersampling are standard methods, our contribution lies in adaptively tuning their parameters based on dataset characteristics. This enhances generalization and classification stability across both datasets.

CIC-MalMem-2022: This dataset contains 58,596 samples across 16 classes (e.g., Emotet, Maze, Zeus). Features are extracted from memory dumps using VolMemLyzer, yielding 55 behavioral attributes such as process activity, file operations, and network connections. Low Variance Filtering is applied to retain only informative features, reducing dimensionality and improving classification efficiency.

B. Experimental Results and Performance Evaluation

This section presents the experimental results and an in-depth analysis of the proposed CMC framework applied to two malware datasets: CMD_2024 and CIC-MalMem-2022. These datasets were selected to comprehensively evaluate the framework's performance under varying conditions, including high-dimensional feature spaces, significant class imbalance, and diverse malware families. The experimental setup and results for each dataset are detailed below.

1. CMD_2024 Dataset

The D2MDM framework (Nguyen et al., 2024 [8]) combines D2MD (Dynamic and Deep Malware Detection) with Random Forest (RF) to improve malware classification. D2MDM integrates CNN, BiGRU, and self-attention mechanisms to capture both local and sequential dependencies in malware data.

In comparison, the CMC framework, which uses Soft Voting to combine DNN and LGBM, shows superior results on the CMD_2024 dataset, achieving 91.79% accuracy and maintaining balanced metrics across precision, recall, and F1 score. The CMC framework excels in classifying imbalanced malware data and efficiently handles high-dimensional datasets, thanks to the use of Soft Voting which leverages the advantages of both deep learning and gradient boosting techniques. The results in Table 2 clearly demonstrate the superior performance of the CMC framework compared to previous approaches on the CMD_2024 dataset.

TABLE 2. COMPARISON OF CMC WITH PREVIOUS APPROACHES ON CMD_2024 DATASET

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
D2MDM (Nguyen et al., 2024 [8])	86.97	87.35	86.97	86.91
CMC Framework	**91.79**	**91.79**	**91.79**	**91.78**

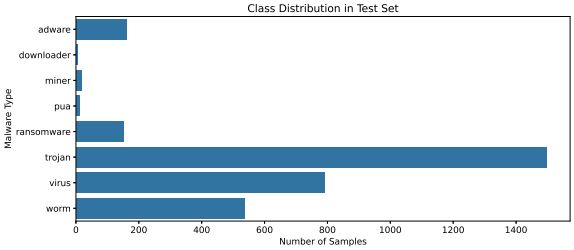
The results in Figure 3 further validate the CMC framework's effectiveness in malware classification. The confusion matrix and ROC curve show strong performance in classifying major malware families like trojans and viruses, while also demonstrating improved accuracy for less common classes like ransomware and miners. The ROC curves show an AUC of 98.1%, confirming the model's ability to robustly distinguish between malware types. These results demonstrate that CMC effectively balances feature selection and model performance, ensuring high accuracy while maintaining computational efficiency.

2. CIC-MalMem-2022 Dataset

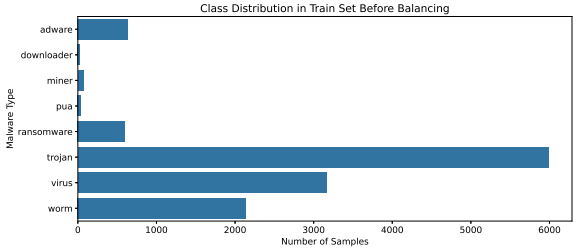
The CIC-MalMem-2022 dataset, specifically designed for memory-based malware detection, contains advanced features derived from memory dumps and system activities, making it particularly suitable for identifying modern threats such as fileless malware. CMC

TABLE 1. CLASS DISTRIBUTION BEFORE AND AFTER BALANCING (SMOTE + RANDOM UNDERSAMPLING) ON TRAINING SET

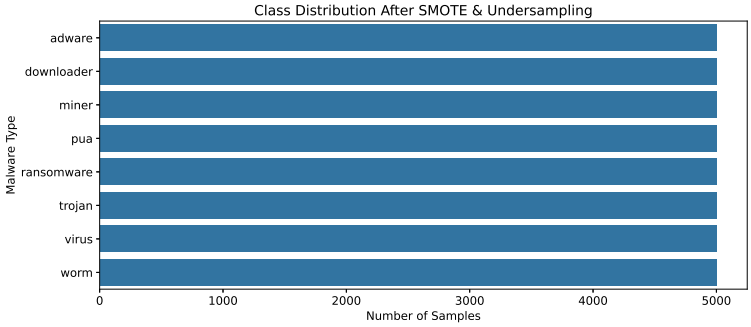
No.	Dataset	Original Data	Train Set Before Balancing	Train Set After Balancing (SMOTE + Random Undersampling)
1	**CMD_2024** (8 Classes)	Trojan: 7484 Virus: 3955 Worm: 2673 Adware: 800 Ransomware: 751 Miner: 90 PUA: 49 Downloader: 32	Trojan: 5987 Virus: 3164 Worm: 2138 Adware: 640 Ransomware: 601 Miner: 72 PUA: 39 Downloader: 26	Trojan: 5000 Virus: 5000 Worm: 5000 Adware: 5000 Ransomware: 5000 Miner: 5000 PUA: 5000 Downloader: 5000
2	**CIC-MalMem-2022** (16 Classes)	Benign: 29298 Transponder: 2410 Gator: 2200 Shade: 2128 CWS: 2000 Scar: 2000 180Solutions: 2000 Ako: 2000 Refroso: 2000 Conti: 1988 Emotet: 1967 Maze: 1958 Zeus: 1950 Pysa: 1717 Recony: 1570 TIBS: 1410	Benign: 13438 Transponder: 1928 Gator: 1760 Shade: 1702 CWS: 1600 Scar: 1600 180Solutions: 1600 Ako: 1600 Refroso: 1600 Conti: 1590 Emotet: 1574 Maze: 1566 Zeus: 1560 Pysa: 1374 Recony: 1256 TIBS: 1128	Benign: 2000 Transponder: 2000 Gator: 2000 Shade: 2000 CWS: 2000 Scar: 2000 180Solutions: 2000 Ako: 2000 Refroso: 2000 Conti: 2000 Emotet: 2000 Maze: 2000 Zeus: 2000 Pysa: 2000 Recony: 2000 TIBS: 2000



(a) Test Set Distribution



(b) Train Set Before Balancing

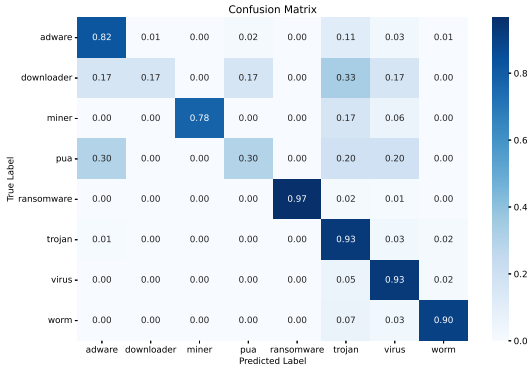


(c) Train Set After Balancing

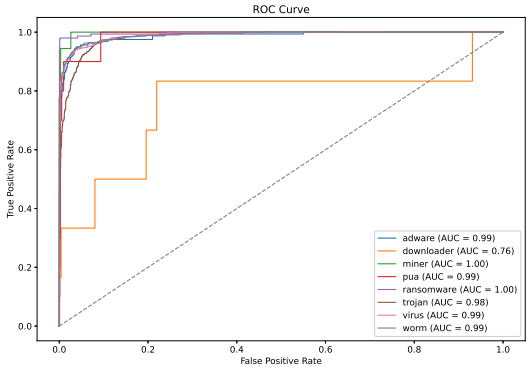
Figure 2. Data Distribution for CMD_2024 (8 Classes)

demonstrated superior performance in the 16-class classification of the CIC-MalMem-2022 dataset. The confusion matrices and ROC curves for both configurations are shown in Figure 4.

The DNN model (Roy et al. [30], 2023) relies

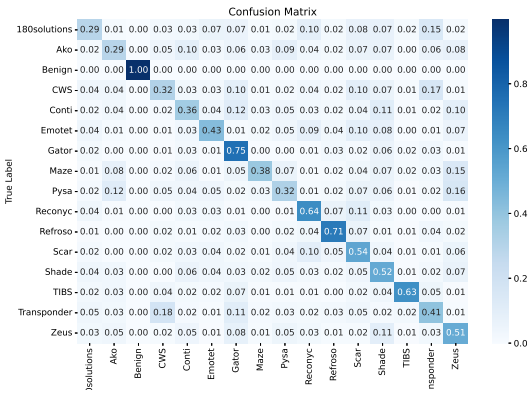


(a) Confusion Matrix

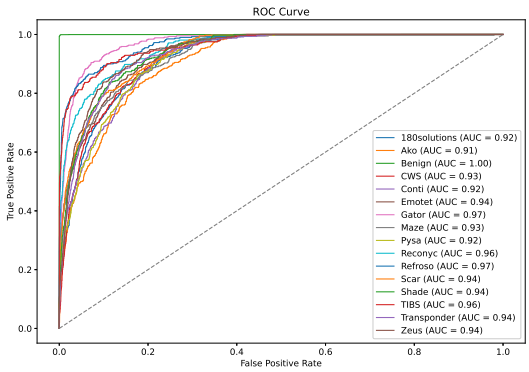


(b) ROC Curve

Figure 3. Evaluation results of CMC on CMD_2024 dataset with 8 classes



(a) Confusion Matrix



(b) ROC Curve

Figure 4. Evaluation results of CMC on CIC-MalMem-2022 dataset with 16 classes

on deep learning techniques to automatically extract features and capture complex, non-linear patterns in the data, making it suitable for malware classification tasks that involve large amounts of dynamic data. DNN has proven effective in many applications, including malware detection, as it is adept at learning from large datasets and detecting intricate patterns in malware behavior.

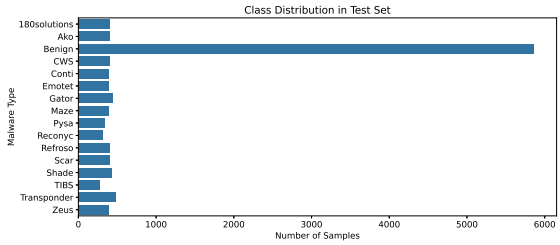
CMC's performance is further validated in the 16-class classification scenario, where it achieved 71.98% accuracy, 71.60% F1-score, and 72.99% precision, as shown in Table 3. These results demonstrate CMC's robust capability to handle multi-class malware classification tasks. The confusion matrix and ROC curve in Figure 5 highlight CMC's ability to minimize misclassifications and demonstrate excellent class separation.

TABLE 3. COMPARISON OF CMC WITH PREVIOUS APPROACHES ON CIC-MALMEM-2022 (16 CLASSES)

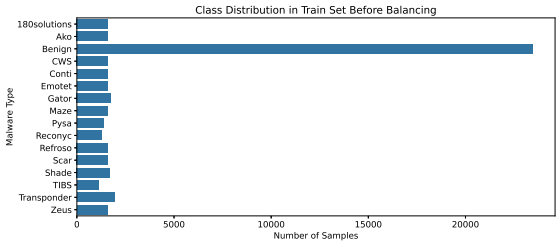
Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
DNN (Roy et al. [30], 2023)	70.29	70.33	72.06	70.33
CMC Framework	**71.98**	**72.99**	**71.98**	**71.60**

Conclusion And Future Work

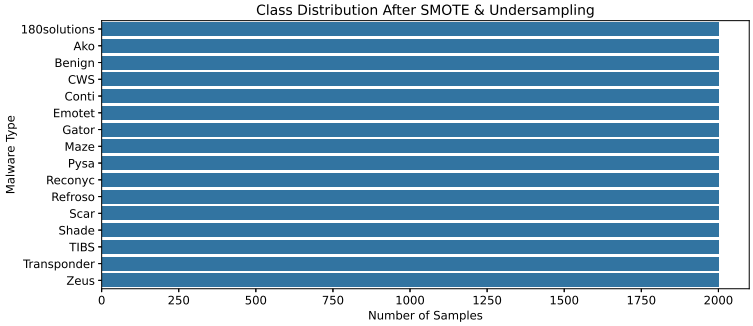
This paper introduces the CMC framework, a robust solution for multi-class malware classification, particularly in cloud environments. By integrating Low Variance Filtering and Scaling, data balancing, dimensionality reduction, and Soft voting-based classification, CMC demonstrates substantial improvements in both classification accuracy and computational efficiency. The experimental results confirm that CMC outperforms existing models on multiple datasets, including highly imbalanced ones such as CMD_2024 and CIC-MalMem-2022.



(a) Test Set Distribution



(b) Train Set Before Balancing



(c) Train Set After Balancing

Figure 5. Data Distribution for CIC-MalMem-2022 (16 Classes)

The CMC framework is tailored for cloud environments, leveraging scalability to dynamically allocate resources for processing large datasets in real-time. It utilizes distributed data processing to monitor and analyze malware behavior across multiple systems, ensuring comprehensive detection. With efficient resource management, CMC optimizes data preprocessing, feature selection, and classification, making it suitable for large-scale deployment. Its cloud-based architecture allows for continuous adaptation to new threats, improving accuracy while balancing cost and performance, particularly in handling imbalanced datasets and high-dimensional features.

A major contribution of this framework is the use of Low Variance Filtering and Scaling in Phase 1, which reduces feature redundancy while preserving essential features for classification. The results indicate that this technique effectively lowers the dimensionality of the datasets, leading to faster prediction times without sacrificing performance. Additionally, the combination of SMOTE and Random Undersampling for data balancing addresses the class imbalance issue, significantly improving detection rates for underrepresented malware families, even in the presence of extreme class imbalance. While CMC shows promising results, there are still challenges to address. One challenge is handling the extreme class imbalance, particularly for emerging and rare malware types. Although the data balancing strategies used in CMC mitigate this issue, future research could explore hybrid sampling methods to enhance this capability further.

REFERENCES

[1] T. Panker, A. Cohen, T. Landman, C. Bery, and N. Nissim, "Mincloud: Trusted and transferable minhash-based framework for unknown malware detection for linux cloud environments," *Journal of Information Security and Applications*, vol. 87, p. 103907, 2024.

[2] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on windows malware detection: Techniques, research issues, and future directions," *Journal of Systems and Software*, vol. 209, p. 111921, 2024.

[3] P. Mishra, T. Jain, P. Aggarwal, G. Paul, B. B. Gupta, R. W. Attar, and A. Gaurav, "Cloudintellmal: An advanced cloud based intelligent malware detection framework to analyze android applications," *Computers and Electrical Engineering*, vol. 119, p. 109483, 2024.

[4] O. Aslan, M. Ozkan Okay, and D. Gupta, "A review of cloud-based malware detection system: Opportunities, advances and challenges," 03 2021.

[5] Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent behavior-based malware detection system on cloud computing environment," *IEEE Access*, vol. 9, pp. 83 252–83 271, 2021.

[6] G. Kale, G. E. Bostancı, and F. V. Çelebi, "Evolutionary feature selection for machine learning based malware classification," *Engineering Science*

and Technology, an International Journal, vol. 56, p. 101762, 2024.

[7] T. Carrier, P. Victor, A. Tekeoglu, and A. Habibi Lashkari, "Detecting obfuscated malware using memory feature engineering," 01 2022, pp. 177–188.

[8] P. S. Nguyen, T. N. Huy, T. A. Tuan, P. D. Trung, and H. V. Long, "Hybrid feature extraction and integrated deep learning for cloud-based malware detection," *Computers & Security*, vol. 150, p. 104233, 2025.

[9] S. Matharaarachchi, M. Domaratzki, and S. Muthukumarana, "Enhancing smote for imbalanced data with abnormal minority instances," *Machine Learning with Applications*, vol. 18, p. 100597, 2024.

[10] H. D. Misalkar and P. Harshavardhanan, "Tdbamla: Temporal and dynamic behavior analysis in android malware using lstm and attention mechanisms," *Computer Standards & Interfaces*, vol. 92, p. 103920, 2025.

[11] C. Li, Z. Cheng, H. Zhu, L. Wang, Q. Lv, Y. Wang, N. Li, and D. Sun, "Dmalnet: Dynamic malware analysis based on api feature engineering and graph learning," *Computers Security*, vol. 122, p. 102872, 2022.

[12] F. H. da Costa, I. Medeiros, T. Menezes, J. V. da Silva, I. L. da Silva, R. Bonifácio, K. Narasimhan, and M. Ribeiro, "Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification," *Journal of Systems and Software*, vol. 183, p. 111092, 2022.

[13] H. juan Zhu, Y. Li, L. min Wang, and V. S. Sheng, "A multi-model ensemble learning framework for imbalanced android malware detection," *Expert Systems with Applications*, vol. 234, p. 120952, 2023.

[14] A. Bensaoud and J. Kalita, "Cnn-lstm and transfer learning models for malware classification based on opcodes and api calls," *Knowledge-Based Systems*, vol. 290, p. 111543, 2024.

[15] R. Chaganti, V. Ravi, and T. D. Pham, "Deep learning based cross architecture internet of things malware detection and classification," *Computers Security*, vol. 120, p. 102779, 2022.

[16] Prachi., N. Dabas, and P. Sharma, "Malanalyser: An effective and efficient windows malware detection method based on api call sequences," *Expert Systems with Applications*, vol. 230, p. 120756, 2023.

[17] S. Kumar and K. Panda, "Sdif-cnn: Stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification," *Applied Soft Computing*, vol. 146, p. 110676, 2023.

[18] S. Yang, Y. Yang, D. Zhao, L. Xu, X. Li, F. Yu, and J. Hu, "Dynamic malware detection based on supervised contrastive learning," *Computers and Electrical Engineering*, vol. 123, p. 110108, 2025.

[19] D. Zhang, Y. Song, Q. Xiang, and Y. Wang, "Imcmk-cnn: A lightweight convolutional neural network with multi-scale kernels for image-based malware classification," *Alexandria Engineering Journal*, vol. 111, pp. 203–220, 2025.

[20] B. B. Gupta, A. Gaurav, V. Arya, S. Bansal, R. W. Attar, A. Alhomoud, and K. Psannis, "Earthworm optimization algorithm based cascade lstm-gru model for android malware detection," *Cyber Security and Applications*, vol. 3, p. 100083, 2025.

[21] M. Alotaibi, G. Aldehim, M. Maashi, M. M. Asiri, F. A. Alrslani, S. R. Alotaibi, A. Yafoz, and R. Alsini, "Chaos game optimization with stacked lstm sequence to sequence autoencoder for malware detection in iot cloud environment," *Alexandria Engineering Journal*, vol. 112, pp. 688–700, 2025.

[22] N. Minh, N. V. Hung, and N. Shone, "Using deep graph learning to improve dynamic analysis-based malware detection in pe files," *Journal of Computer Virology and Hacking Techniques*, vol. 20, pp. 1–20, 10 2023.

[23] P. V. Dinh, N. Shone, P. H. Dung, Q. Shi, N. V. Hung, and T. N. Ngoc, "Behaviour-aware malware classification: Dynamic feature selection," in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 2019, pp. 1–5.

[24] A. Çayır, U. Ünal, and H. Dağ, "Random capsnet forest model for imbalanced malware type classification task," *Computers Security*, vol. 102, p. 102133, 2021.

[25] F. Demirkıran, A. Çayır, U. Ünal, and H. Dağ, "An ensemble of pre-trained transformer models for imbalanced multiclass malware classification," *Computers Security*, vol. 121, p. 102846, 2022.

[26] L. Xue and T. Zhu, "Hybrid resampling and weighted majority voting for multi-class anomaly detection on imbalanced malware and network traffic data," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107568, 2024.

[27] M. A. Latif, Z. Mushtaq, S. Rahman, S. Arif, S. Mursal, M. Irfan, and H. Aziz, "Oversampling-enhanced feature fusion-based hybrid vit-1dcnn model for ransomware cyber attack detection," *Computer Modeling in Engineering Sciences*, vol. 142, pp. 1667–1695, 01 2025.

[28] N. Anđelić, S. Baressi Šegota, and V. Mrzljak, "Application of symbolic classifiers and multi-ensemble threshold techniques for android malware detection," *Big Data and Cognitive Computing*, vol. 9, pp. 1–49, 01 2025.

[29] M. A. R. Putra, T. Ahmad, D. P. Hostiadi, and R. M. Ijtihadie, "Ensemble network graph-based classification for botnet detection using adaptive weighting and feature extraction," *IEEE Access*, vol. 13, pp. 31 183–31 204, 2025.

[30] K. S. Roy, T. Ahmed, P. B. Udas, M. E. Karim, and S. Majumdar, "Malhystack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis," *Intelligent Systems with Applications*, vol. 20, p. 200283, 2023.

ABOUT THE AUTHORS

Pham Sy Nguyen
Workplace: Cyber Security Center, Office of the Government, Vietnam
Email: phamsynguyen@chinhphu.vn
Education: Graduated with a Engineering's degree in Information Security from the Vietnam Academy of Cryptography Techniques, Hanoi, Vietnam in 2013, and received an MSc in Information Security from the same institution in 2016.
Recent research interests: He is currently concerned in the fields of Machine Learning, Deep Learning, Artificial Intelligence and applications in Cybersecurity.

Tên tác giả: **Phạm Sỹ Nguyên**
Cơ quan công tác: Văn phòng Chính phủ
Email: phamsynguyen@chinhphu.vn
Quá trình đào tạo: Tốt nghiệp Kỹ sư chuyên ngành An toàn thông tin tại Học viện Kỹ thuật mật mã, Hà Nội vào năm 2013, và nhận bằng Thạc sĩ cùng chuyên ngành tại chính học viện này vào năm 2016.
Hướng nghiên cứu hiện nay: Học máy, học sâu, trí tuệ nhân tạo và ứng dụng trong lĩnh vực an toàn thông tin.

Pham Ngoc Van
Workplace:Information Security Center, Military Commercial Joint Stock Bank (MBBank), Vietnam
Email: phamngocvan.kma@gmail.com
Education: Received his Engineering degree and M.Sc. in Information Security from the Vietnam Academy of Cryptography Techniques, Hanoi, Vietnam, in 2020 and 2023, respectively.
Recent research interests: Applying Machine Learning, Deep Learning, and Artificial Intelligence in cybersecurity, especially malware detection, threat prevention, and enhancing system defenses.

Tên tác giả: **Phạm Ngọc Vân**
Cơ quan công tác: Trung tâm An toàn thông tin, Ngân hàng Thương mại Cổ phần Quân đội (MBBank)
Email: phamngocvan.kma@gmail.com
Quá trình đào tạo: Tốt nghiệp Kỹ sư (2020) và Thạc sĩ (2023) chuyên ngành An toàn thông tin tại Học viện Kỹ thuật mật mã, Hà Nội.
Hướng nghiên cứu hiện nay: Ứng dụng học máy, học sâu và trí tuệ nhân tạo trong an toàn thông tin, tập trung vào phát hiện mã độc, ngăn chặn mối đe dọa và tăng cường phòng thủ hệ thống.

Hoang Viet Long
Workplace: Faculty of Information Technology and Cybersecurity, University of Technology - Logistics of Public Security, Vietnam
Email: longhv08@gmail.com
Education: Received PhD diploma in Computer Science at Hanoi University of Science and Technology in 2011, specializing in fuzzy computing and soft computing techniques with applications to electronics engineering. He has been promoted to Associate Professor since 2018.
Recent research interests: Mathematical modeling, Machine Learning, Deep Learning, and their applications in Cybersecurity.

Tên tác giả: **Hoàng Việt Long**
Cơ quan công tác: Khoa Công nghệ Thông tin và An ninh mạng, Trường Đại học Kỹ thuật - Hậu cần Công an Nhân dân
Email: longhv08@gmail.com
Quá trình đào tạo: Nhận bằng Tiến sĩ chuyên ngành Khoa học máy tính tại Trường Đại học Bách Khoa Hà Nội năm 2011, chuyên sâu về fuzzy computing và soft computing trong kỹ thuật điện tử. Được phong học hàm Phó Giáo sư từ năm 2018.
Hướng nghiên cứu hiện nay: Mô hình toán học, học máy, học sâu và ứng dụng trong an toàn thông tin.

Pham Duy Trung
Workplace: Faculty of Information Security, Academy of Cryptography, Vietnam
Techniques Email: trungpd@actvn.edu.vn
Education: Received PhD diploma in Information Systems at University of Canberra, Australia in 2018, specializing watermarking approach for data protection.
Recent research interests: Recently, he has concerned in adversarial attacks, deep learning with application in Cybersecurity.

Tên tác giả: **Phạm Duy Trung**
Cơ quan công tác: Khoa An toàn Thông tin, Học viện Kỹ thuật mật mã
Email: trungpd@actvn.edu.vn
Quá trình đào tạo: Nhận bằng Tiến sĩ chuyên ngành Hệ thống Thông tin tại Trường Đại học Canberra, Úc vào năm 2018, chuyên sâu về phương pháp watermarking trong bảo vệ dữ liệu.
Hướng nghiên cứu hiện nay: Các cuộc tấn công đối kháng (adversarial attacks), học sâu và ứng dụng trong lĩnh vực an toàn thông tin.