MDPI

Article

Hierarchical Intrusion Classification using Machine Learning on a Consolidated Benchmark Dataset

Yunji Kim [1,†], Junghye Kim [2,†], Jihyeon Kim [3,†] and Dongho Kim [4,*]

[1] Dept. of Artificial Intelligence, Dongguk University, Seoul, Korea; 2019112107@dgu.ac.kr
[2] Dept. of Information and Communication Engineering, Dongguk Univ., Seoul, Korea; tweetiijh@dgu.ac.kr
[3] Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea; jiwlgus048@dongguk.edu
[4] Software Education Institute, Dongguk University, Seoul, Korea; dongho.kim@dgu.edu
* Correspondence: dongho.kim@dgu.edu
† These authors contributed equally to this work.

Abstract: Given the risk of intrusion and misuse inherent in data exchange over the network, developing proper countermeasures by identifying specific intrusions to protect the system's integrity is critical. This paper presents a novel hierarchical multi-step intrusion detection and classification model based on machine learning. Rather than categorizing malicious and benign network traffic simultaneously, our approach initially segregates the traffic into malicious and benign ones. Subsequently, we classify only the malicious traffic into four groups before pinpointing them as specific attack types. We merged two popular packet capture datasets to address various intrusion scenarios effectively, then trimmed their size to facilitate lightweight classification. After employing only essential features and selecting the top-performing models, our hierarchical approach demonstrated remarkable results, yielding up to 151.7% improvement in classification performance compared to conventional flat classification. An impressive recall rate of 98.8% highlighted this achievement.

Keywords: Network Intrusion Detection; Hierarchical Classification; Data Format Conversion; Machine Learning; Feature Selection

1. Introduction

In the realm of cybersecurity, Intrusion Detection Systems (IDS) play a crucial role in safeguarding network infrastructures by identifying malicious activities and potential security breaches. These systems act as vigilant guardians, continuously monitoring and analyzing network traffic to detect anomalies and suspicious patterns that may indicate an ongoing or impending cyber intrusion. Therefore, this study will focus on developing a more efficient and lightweight classification model that can help real-time detection[1].

Packet Capture (PCAP) data, encompassing a record of all system interactions, serves as a comprehensive information source in IDS[2]. However, its sheer volume and resource-intensive nature render it unsuitable for real-time detection due to processing time and storage capacity constraints. Machine Learning (ML) models, particularly those suited for signature-based intrusions, require significant diverse data with useful information for effective training. Learning from data like PCAP, which has unclear informational value and is time-consuming, is challenging. To overcome this, we explored benchmark datasets encompassing various intrusion types and rich in valuable information. Therefore, we decided to utilize CIC-IDS 2017[3] and UNSW-NB15[4], which provide diverse intrusion types and effectively extract useful information from PCAP data.

Integrating these datasets has allowed us to address various types of intrusions. However, overlapping intrusion types were discovered during the merging of the two datasets, resulting in classification errors. Additionally, class imbalance issues were identified, where malicious intrusions were significantly underrepresented compared to benign data, or certain intrusion types were overly prevalent. This imbalance in the data posed challenges

Citation: Kim, Y.; Kim, J.; Kim, J.; Kim, D. Hierarchical Intrusion Classification using Machine Learning on a Consolidated Benchmark Dataset. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

in effectively classifying less frequent intrusion types. To address these issues, we proposed a hierarchical ML model architecture. Initially, classifying intrusions into broad categories and further subclassifying them into specific categories showed promising performance in situations where data imbalance existed. Furthermore, to enhance real-time detection, we encouraged feature selection to streamline the necessary information collection. Various feature selection techniques were devised based on encountered model construction errors.

In summary, in this study:

1. Preprocess the PCAP data from the existing two datasets for network intrusion detection according to the methods in the respective papers and create two datasets that include more intrusion types for each data format.
2. Considering that benign traffic is predominant in typical situations, we propose a hierarchical model structure that can accurately classify the specific intrusion type.
3. Select essential features for each data format and construct a lightweight system for real-time and efficient processing of various intrusion types and traffic.

The rest of the paper proceeds as follows: Section 2 discusses previous research on NIDS systems and studies involving CIC-IDS 2017 and UNSW-NB15 datasets. Section 3 explains the dataset creation and preprocessing methods, hierarchical model structure, and feature selection. Section 4 presents experimental results demonstrating the advantages of the hierarchical model structure and provides a list and analysis of the selected features. Section 5 offers a discussion of future research. Finally, Section 6 summarizes this study and describes future work.

2. Related Works

The methodologies for intrusion detection systems are classified into three categories: Signature-based Detection (SD), Anomaly-based Detection (AD), and Stateful Protocol Analysis (SPA)[5]. Based on Signature-based Detection, existing research on using machine learning to build intrusion detection systems has trained models based on network traffic analysis. It has proposed the importance of network traffic analysis, the necessity of machine learning models, and the significance of a centralized network sampler.

2.1. Selecting and Preparing Datasets for Network Traffic Analysis

In [6], various datasets that can be used for training network intrusion detection models were compared and analyzed. It suggested datasets such as CIC-IDS2017, DARPA, KDD CUP 99, and UNSW-NB15. In this study, we examined various datasets like those mentioned, and we decided to use the CIC-IDS2017 and UNSW-NB15 datasets, which provide packet data for research. [3] captured all incoming and outgoing traffic on the main switch of the victim network's major switches and generated packet capture files for the CIC-IDS2017 dataset. The raw logs in PCAP format can be reconstructed into flow-based datasets using the CICFlowMeter sensor proposed in the paper. The CIC-IDS2017 dataset includes various intrusion types such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan, and Botnet intrusions. [7], [8], [9] have pointed out and improved errors in CIC-IDS2017.

[4] connected each server through a router and captured network traffic from the router to create a packet capture file of UNSW-NB15. The PCAP file can be reconstructed into a Flow-based Dataset through the Argus and Bro Sensors proposed in the study. The UNSW-NB15 dataset contains intrusion types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. CIC-IDS2017 is a data set related to flow time-related statistics and contains information about statistics measured for a fixed time of 1 AT and 1 second. UNSW-NB15 contains connection-related information and flow sequence-related information.

In this study, we constructed integrated datasets to create a unified dataset that shares more features and includes a broader range of intrusion types. We applied this suitable

dataset to a single model. Previous research that evaluated a single model using multiple datasets consists of the following: [10] introduced the concept of an anomaly detection framework that can perform intrusion detection using data from various aspects frequently used recently. [11] presented the idea of a hybrid IDS that combines signature-based IDS and anomaly-based IDS. Engineering models based on Information Gain (IG), Fast Correlation-based Filter (FCBF), and Kernel Principal Component Analysis (KPCA) algorithms were proposed, as well as an IDS based on CL-k-means. In this research, we converted and merged the PCAP files of both datasets into datasets with the same features found in their generation methods. Data converted into the CIC-IDS2017 format integrated into the CM-CIC-IDS2017 dataset, and data converted into the UNSW-NB15 format integrated into the CM-UNSW-NB15 dataset. We created CM-CIC-IDS2017 and CM-UNSW-NB15 datasets in the CIC-IDS2017 format for this study.

2.2. Model and System Architecture for Intrusion Detection Systems

In [12], [13], [14], DBN (Deep Belief Networks), RF (Random Forest), and DFNN (Deep Feedforward Neural Network) Specification Models were each selected, and research was conducted to compare the performance with several detection models. [15] proposed a data-driven system for active data monitoring and used an algorithm based on distance measures of the empirical cumulative distribution function (ECDF). These studies trained models based on network traffic analysis. They provided implications for network traffic analysis, suggesting the need for an AI model and the importance of a centralized Network Sampler. According to existing network traffic detection studies, it is crucial to show accuracy using AI models, and it is necessary to build a system that can continuously monitor.

[16] proceeded with multi-class classification using a hierarchical intrusion detection system. This study constructed a hierarchical structure to classify various intrusion types of the integrated dataset.[17] implemented the intrusion detection system architecture proposed in the paper by incorporating it into the Zeek open-source project. This idea can be used when building a network intrusion detection system by directly collecting data in the future.

2.3. Feature Selection

Studies focusing on feature importance include [18], considered complex features representing sophisticated intrusions to improve the accuracy of traffic anomaly detection. Information gain was used to rank relevant features and group them by weight. They proposed a method for determining a feature set effective for a specific intrusion by substituting grouped features into a classifier algorithm and analyzing the results. [3] extracted 80 features from the PCAP file using CICFlowMeter, which extracted flow-based features for the CIC-IDS2017 dataset and classified them using the RF class. In this process, They proposed a method to extract the best feature set for each intrusion type detection by calculating each feature's importance, the average standardized mean value of each feature, and the corresponding feature importance in each class. [19] proposed a flow-based classification method that can characterize encrypted and VPN traffic using only time-related features. When choosing time-related features, They used two approaches: one that measures only time and one that fixes time and measures other variables. Two scenarios were used, one in which traffic is characterized by classifying it into VPN and non-VPN, and the other in which traffic is described without distinguishing between VPN and non-VPN. Using the C4.5 and KNN algorithms, They proved that time-related features are good classifiers.

3. Methodology for Efficient Lightweight Intrusion Classification

In this section, we elaborate on how we devised our intrusion detection model to be accurate and lightweight, as shown in Figure 1. First, in 3.1, we describe how we performed format conversion of packet capture data to procure more data, obtain more intrusion types,

and lighten the dataset for real-time processing. Next, in 3.2, we explain how we designed a hierarchically structured classification model to detect various intrusions accurately. Finally, in 3.3, we elucidate the feature selection process to reduce the information to be transformed and formulate a lighter classification model.

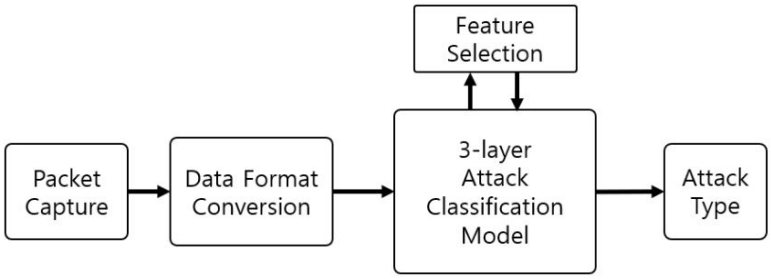

Figure 1. Hierarchical Intrusion Classification Architecture with Dataset Consolidation .

3.1. Dataset Consolidation for more Data and Information

3.1.1. CIC-IDS2017 and UNSW-NB15 Data Format

The CIC-IDS2017 and UNSW-NB15 datasets are recognized as key benchmarks in modern network security research, each reflecting a variety of intrusion detection scenarios on the network. While these datasets provide detailed network traffic information, they are distinguished by their unique collection and processing methods. CIC-IDS2017 is provided by the Canadian Institute for Cybersecurity(CIC) and is designed to capture a variety of intrusion patterns under complex network conditions. This dataset includes 7 primary intrusion types and 14 detailed intrusion types, as well as benign traffic. UNSW-NB15, on the other hand, was developed by the Australian Centre for Cyber Security (ACCS) and aims to capture a variety of intrusion patterns over a shorter collection period. This dataset provides a mix of network traffic, including 9 primary intrusion types as well as benign traffic. Both datasets utilize unique attribute extraction tools to extract valuable information from network traffic. CIC-IDS2017 used CICFlowMeter and UNSW-NB15 used Argus and Bro-IDS tools to extract features. Table 1 summarizes the key features of the CIC-IDS2017 and UNSW-NB15 datasets.

Table 1. Organization of the Datasets

Dataset Name	**CIC-IDS2017**	**UNSW-NB15**
Provider	Canadian Institute for Cybersecurity(CIC)	Australian Centre for Cyber Security (ACCS)
Dataset Contents	Mixture of benign traffic, 7 primary intrusion types and 14 detailed intrusion types	Mixture of benign traffic and 9 intrusion types
Data Collection Period	Five days	Two days
Data Size	51.1 GB of raw data in PCAP format	100 GB of captured raw traffic
Features	84 flow-based features were extracted using CICFlowMeter software[20]	49 features were extracted from Argus and Bro-IDS tools[4]

3.1.2. Dataset Consolidation Process

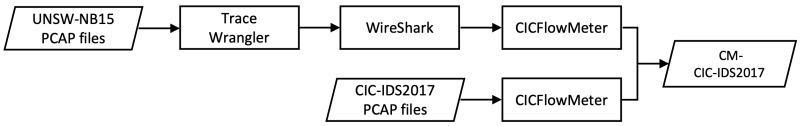

Figure 2. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-CIC-IDS2017

Figure 2 depicts combining the UNSW-NB15 and CIC-IDS2017 PCAP files to produce CIC-IDS2017 formatted data using CICFlowMeter[20]. The link layer protocol header must be replaced to convert the UNSW-NB15 PCAP data through CICFlowMeter, as CICFlowMeter only processes packets in the Ethernet header. In contrast, UNSW-NB15's link layer protocol header is Linux Cook Mode Capture (SLL). TraceWrangler software is used to convert this Linux-cooked header to an Ethernet header. Since the link layer header is not the data range of interest, it is acceptable to use a pseudo-MAC address to convert it to an Ethernet header. Several erroneous packets are split and removed using Wireshark Splitcap, combined into PCAP files using Wireshark Mergecap, and applied to the Trace Wrangler software. The preprocessed data is converted into CIC-IDS2017 formatted data using CICIFlowMeter. Packet data from UNSW-NB15 and CIC-IDS2017, where 84 features were extracted and modified through CICIFlowMeter, is returned as data in CIC-IDS2017 format in CSV output. We name this consolidated dataset as ***CM*-CIC-IDS2017**.

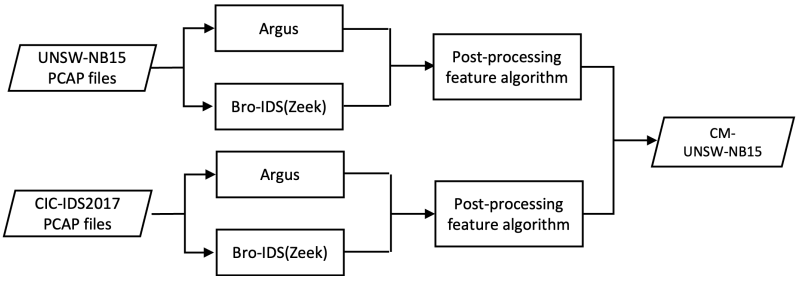

Figure 3. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-UNSW-NB15

Figure 3 illustrates the merging procedure of the CIC-IDS2017 packet data file and UNSW-NB15 packet data file, which undergoes a 3-step transformation to achieve the UNSW-NB15 format. Throughout this process, the model employs the Argus tool to generate main features, utilizes Bro-IDS to generate additional features, combines the two datasets using Flow ID, and subsequently generates the remaining features through post-processing[4]. The Argus tool processes network packet data in PCAP format to create bidirectional flow data, employing both argus-server and argus-client components. The former records input PCAP files in binary format within the argus file format, while the latter extracts features from these argus files. Bro-IDS, an open-source network traffic monitor, generates features through three distinct logs: the conn log records essential connection information, the HTTP log captures request and reply details, and the FTP log encompasses all FTP protocol-related activities. By integrating these log files, the desired features are extracted. The features generated by these two tools are merged using the Flow 5-tuple feature, with the Argus feature containing flow-based information and the Bro feature consisting of packet-based features. This consolidated data undergoes a post-processing algorithm [4], generating 11 supplementary features. Forty-nine features are extracted from the UNSW-NB15 and CIC-IDS2017 PCAP data, subsequently provided as UNSW-NB15 format data in CSV output format. We name this consolidated dataset as ***CM*-UNSW-NB15**.

3.1.3. Data Size Reduction for Lightweight Processing

PCAP data captures information about every packet sent on a network, providing a detailed snapshot of network activity. Because of the depth of detail in this data, direct analysis or processing of PCAP files requires large computing resources and memory. To efficiently manage and process the high volume and detail of PCAP data, it is essential to convert the data to another format. This conversion process can reduce the size of the data while improving the efficiency of analysis and processing. Table 2 shows the results of converting PCAP data to other data formats. The size of the data after conversion and the resulting percentage reduction in size provide a clear picture of the importance and effectiveness of data conversion.

Table 2. Data Size Reduction after Format Conversion

	CIC-IDS2017		UNSW-NB15		**Merged**	
PCAP Dataset(GB)	47.90		50.20		**98.10**	
	Size(GB)	Reduction(%)	Size(GB)	Reduction(%)	Size(GB)	Reduction(%)
CICIFlowmeter	1.30	97.29	1.60	96.81	**3.00**	**96.64**
Argus with Bro	0.48	99.01	0.48	99.05	**0.95**	**99.03**

Assuming a data collection period of 30 days with five days of actual data collection, the CIC-IDS2017 dataset amounted to 287.40GB. After conversion into each dataset format, it was reduced to 8.08GB for the CIC-IDS2017 format and 2.86GB for the UNSW-NB15 format. Similarly, assuming a data collection period of 30 days with two days of actual data collection, the UNSW-NB15 dataset was 753.00GB. After conversion into each dataset format, it was reduced to 24.73GB for the CIC-IDS2017 format and 7.45GB for the UNSW-NB15 format. When both datasets collected over 30 days are combined and converted, the total size reaches 1,040.40GB. Depending on the conversion method, the sizes are reduced between 34.17GB and 10.40GB.

The results demonstrate that converting PCAP data into the CIC-IDS2017 or UNSW-NB15 format significantly reduces storage requirements. This conversion improves the model's performance and aids in efficient storage management.

3.1.4. Preprocessing for Machine Learning: Encoding and Scaling

To perform a machine learning model, the data consisting of characters needs to be encoded into numbers, and the distribution of the numbers needs to be adjusted through a scaling process. Therefore, categorical values were encoded using one-hot encoding, and numeric values were transformed using Quantile Transformer to conform to the input format of the model. The features obtained through the CIC-Flowmeter transformation method are all numeric values. In contrast, the features obtained through the Argus with Bro transformation method include categorical values such as 'proto', 'state', and 'service'; the rest are all numeric. The labels indicating the type of intrusion and whether an intrusion occurred were labeled using Label Encoding.

3.2. Hierarchical Intrusion Classification with Machine Learning

By combining the CIC-IDS2017 dataset and the UNSW-NB15 dataset, we obtained 23 intrusion types. To effectively classify various intrusion types, we designed a hierarchical structure. Layer 1 detects malicious activities, layer 2 categorizes the malicious into four groups, and layer 3 specifically classifies into 23 attack types. Ten machine-learning models with hyperparameter tuning processes each classification layer.

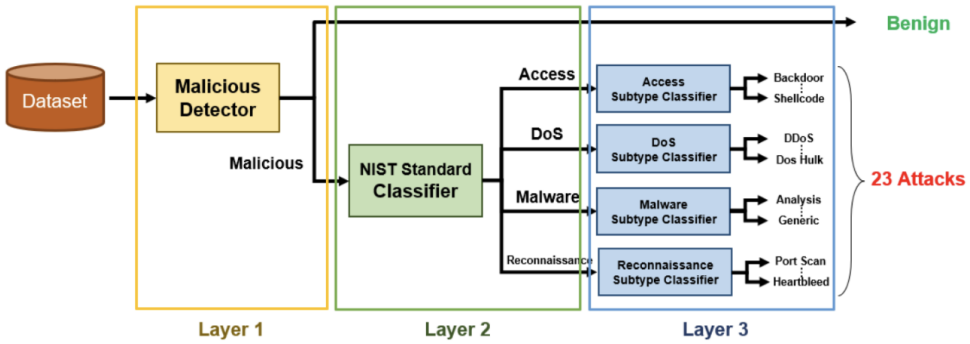

Figure 4. Hierarchical Intrusion Classification Framework/Pipeline

3.2.1. Hierarchical Architecture: Three Layers

To compare the performance of non-hierarchical and hierarchical models for intrusion detection and classification, a hierarchical model was constructed using three layers. The above model can be easily adapted by adding or modifying layers to accommodate new intrusion types and evolving threats.

As shown in Figure 4, Layer-1 functions as a malicious detector, detecting whether the traffic is malicious. The model predominantly learns to differentiate between malicious and benign traffic. Layer-2 operates as an NIST standard classifier, categorizing malicious traffic into four categories: Access, DoS, Malware, and Reconnaissance. This classification adheres to the NIST standard outlined in Table 3, sorting similar intrusion types initially among the 23 intrusion types. Layer-3, within the previously classified four intrusion categories, a further subdivision into 23 specific intrusion types takes place. The model was trained to discern subtle differences among similar intrusions in more detail.

Table 3. Layer-2 Intrusion Groups Categorized by NIST Standard

NIST Category	Intrusion Type	Description
Reconnaissance	PortScan, Web Attack-Brute force, Web Attack-XSS, Web Attack-sql injection, heartbleed, Reconnaissance	These intrusions involve the gathering of information about a computer system or network to identify vulnerabilities that can be exploited in a future intrusion. This can include intrusions such as port scanning, network mapping, and OS fingerprinting.
Access	FTP-Patator, SSH-Patator, Bot, Infiltration, Backdoor, Shellcode, Exploits, Fuzzers, Worms	These intrusions are designed to gain unauthorized computer system or network access. This can include intrusions such as password guessing, social engineering, and exploiting vulnerabilities in software or hardware.
DoS	DoS, DoS Hulk, DDoS, DoS GoldenEye, DoS slowloris, DoS Slowhttptest	These intrusions are designed to overwhelm a network or computer system with traffic or requests, making it unavailable to legitimate users. This can include intrusions such as flooding a network with traffic or exploiting vulnerabilities in network infrastructure.
Malware	Generic, Analysis	These intrusions involve using malicious software to compromise a computer system or network. This can include intrusions such as viruses, worms, and Trojan horses.

3.2.2. Machine Learning Models for Signature-Based Network Intrusion Classification

The CM-CIC-IDS2017 and CM-UNSW-NB15 datasets primarily focus on commonly known types of intrusions, making them closer to signature-based network intrusions. Machine learning models are widely used in various fields, especially for data-driven pattern

recognition and classification tasks, showcasing excellent performance. Thus, using machine learning models to classify intrusion types of signature-based network intrusions can be highly effective. Machine learning models have relatively simple structures requiring less computation and shorter training time than deep learning models. Machine learning models are known for their interpretability, enabling explanations for why classification results are obtained. We selected the best model among ten popular machine learning classification models - Decision Tree[21], Random Forests[22] Gaussian Naive Bayes[23], Linear Discriminant Analysis[24], Quadratic Discriminant Analysis[srivastava2007bayesian], Logistic Regression[25], AdaBoost Classifier[26], K-Neighbors Classifier[27], Multi-Layer Perceptron Classifier[28], Support Vector Machine[29]. Detailed explanations for each model are in Appendix A, Table 1[30].

3.2.3. Hyperparameter Optimization

We utilized GridSearchCV[31] to fine-tune the performance of our machine learning models by finding the most effective combination of hyperparameters. GridSearchCV systematically explores a predefined hyperparameter grid, essential for optimizing the model's performance. For every model in our architecture, we crafted hyperparameter grids, tailoring them to the specific needs of each machine learning model. These grids encompass crucial parameters necessary for model optimization, and the detailed table showcasing these Hyperparameter Grids can be referenced in Appendix A Table 2[30].

3.3. Feature Selection for Lightweight Model

In this paper, we propose a new feature selection method through dual perspective analysis based on the feature importance score calculated by a machine learning model. The CM-CIC-IDS2017 dataset has 66 features and the CM-UNSW-NB15 dataset has 208 features. With such a large number of features, it can take a long time for the model to understand and analyze the data, and unnecessary information can confuse the prediction. In real-world network situations, if the speed of analysis is slower than the speed of incoming packets, the delay in analysis can be fatal to detecting and responding to threats. Therefore, we want to reduce the time it takes for the model to predict intrusions. First, we utilized the XGBoost algorithm to derive the importance score of each feature across multiple layers. Based on this evaluation, we finally make decisions about feature selection in two ways : Feature Influence Analysis and Feature Similarity Analysis to reduce the misclassification rate. We focus on the results in Section 3.3.1, but refer to the results in Sections 3.3.2 and 3.3.3 to make our final decision on feature selection.

3.3.1. Feature Importance Analysis by Ensemble Learning

XGBoost model approach utilizes an ensemble learning method involving multiple decision trees[32]. When computing feature importance, the model selects features based on information gained in each tree to partition the data effectively. This enables an understanding of the importance of each feature, with higher information gain indicating a more significant influence on predictions. Feature Importance is measured based on the number of times a feature is used for partitioning or its Information Gain. It serves as a tool for model interpretation and feature selection.

3.3.2. Feature Similarities between Malicious Intrusion Types

Upon analyzing the classification errors, we observed a tendency for specific malicious traffic to either cluster into a single type of intrusion or be classified as similar but distinct intrusions. Consequently, we devised a method to address this by calculating the similarities among network intrusions for each feature and removing features that poorly classify intrusions or exhibit high similarity between intrusion types. During Exploratory Data Analysis (EDA), we discovered that many features follow a normal distribution centered

around the mean. Hence, we calculated the distance between feature means for each intrusion type, considering it as a measure of similarity between intrusions.

$$FI = 1 - \frac{\sum_i^M \sum_j^M (1 - |\mu_i - \mu_j|)}{M^2} \quad (1)$$

where:

FI = Feature Importance
M = Number of intrusion types, which is 23 in our CM datasets
i, j = Intrusion type index
μ_i = The average of each feature data that belongs to the i-th intrusion type

The formula calculates the Feature Importance (FI) based on intrusion-wise similarity. It calculates each intrusion, deriving the difference between any two intrusion types. The difference ranges from 0 to 1, with smaller differences indicating similarity. The value obtained from the difference is subtracted from 1 to represent this similarity. The resulting values are then summed for all intrusion combinations, generating a similarity matrix for intrusion combinations, and the total sum is calculated. If all intrusions are considered to be the same, the sum of the matrix's diagonal elements would be M^2. Therefore, the total sum obtained is normalized from 0 to 1 by dividing it by M^2. Additionally, since FI should increase when deemed dissimilar, the current calculated values are subtracted from 1.

3.3.3. Feature Influence Analysis to Reduce Misclassification

In this section, we backtracked the intrusion type classification results to analyze the intrusion types with high misclassification rates and analyze what causes them to be misclassified as other types. From our simulations, the intrusion types, including Fuzzers, Analysis, Backdoor, DoS, Exploits, Reconnaissance, Shellcode, and Worms exhibited a misclassification rate exceeding 50%.To address this, we adopted the following analytical strategy. First, We partitioned the data of these problematic intrusion types into the ground truth and mismatched groups. Ground truth group consists of instances where the predicted type matches the actual type. Conversely, the mismatched group comprises instances where the predicted type does not align with the actual type. We aim to identify features that reflect the discrepancy group as closely as possible to the ground truth group. This involved calculating the mean value for each feature within both groups and subsequently discerning disparities in these mean values between the groups. Features exhibiting significant mean differences were identified as potential causes for the discrepancies between the two groups, thereby contributing to the high misclassification rate. Therefore, features with pronounced differences, believed to be the main contributors to misclassification, were excluded to enhance classification accuracy.

4. Experiments and Evaluation

This section provides the experimental setup and evaluation procedures. We supply an overview of the dataset and evaluation configurations, followed by a detailed review of the framework's performance. Subsequently, we examine the feature selection process.

4.1. Consolidated Dataset

4.1.1. Two CM Datasets: Configuration

The dataset consists of 23 distinct types of intrusions that are divided into four categories based on the NIST standard. These categories include Reconnaissance, Access, Denial of Service (DoS), and Malware, depending on which instances they appear and how often they occur. We converted the CM-CIC-IDS2017 and CM-UNSW-NB15 datasets into intrusion type configurations to facilitate the training and evaluation of machine learning models. When partitioning the dataset, we implemented stratified sampling based on four columns divided by intrusion category. As a result, our training and test datasets were split

into a 7:3 ratio. A detailed description of the intrusion categories for each dataset and the training dataset composition is provided in Appendix B[33].

4.1.2. Evaluation Criteria: False Negative Rates

We selected the recall metric as our primary evaluation criterion. This choice was rooted in the significance of accurate intrusion detection, particularly the importance of minimizing false negative(FN) rates, since not missing malicious incidents is critical. Recall measures the proportion of correctly predicted positive(intrusion) instances among all actual positive instances, providing a clear assessment of the model's ability to identify intrusions. It can be expressed as follows:

$$Recall(TruePositiveRate) = \frac{TP}{TP + FN} \tag{2}$$

As evident from the formula, enhancing recall entails reducing false negative rates. This aligns well with the imperative of minimizing missed detections in intrusion scenarios. Consequently, we employed recall as the central metric in our evaluation process.

For the multi-class classification scenario, we leveraged the scikit-learn library, setting the average option. Given the inherent class imbalance with many benign instances, we adopted the weighted average approach to address this disparity. This methodology assigns appropriate weights to each label, compensating for the skewed distribution of instances. Our study underscores the importance of prioritizing the reduction of false negative rates and utilizing suitable evaluation metrics to ensure accurate intrusion detection.

4.2. Assessment of the Hierarchical Multi-step Framework

This section presents our novel hierarchical intrusion detection models obtained from the layered approach introduced in Section 3.2.1. The hierarchical 2-step model and the hierarchical 3-step Model are each designed to enhance intrusion detection precision through layered architectures

The first model adopts a 2-step approach. Layer-1 classifies incoming data into benign or malicious categories, while Layer-3 refines the process by categorizing malicious data into 23 distinct intrusion types. The second model, which extends the hierarchy into a 3-step structure, integrates three layers. Like the first model, data is initially categorized as benign or malicious in Layer-1. Layer-2 proceeds to classify malicious data into four primary intrusion categories. Finally, Layer-3 classifies data into sub-intrusion types within each category.

To investigate the impact of adding layers on performance, we compared a 1-step (non-hierarchical) model which only uses Layer-3 with a hierarchical 2-step model consisting of two layers and a hierarchical 3-step model consisting of three layers.

4.2.1. Optimal Model Selection

A process was undertaken to select the optimal model at each layer to enhance the overall performance of intrusion classification. To accomplish this task, we utilized the GridSearchCV library from scikit-learn to search for the optimal hyperparameters while selecting the best-performing model.

The optimal model was selected separately for each layer. The tables below present the chosen optimal models for each layer and their respective recall values.

Table 4. Performance of the best model for each layer and classifier of the hierarchical 2-step model.

	CM-CIC-IDS2017		**CM-UNSW-NB15**	
	Model	**Recall**	**Model**	**Recall**
Layer-1	RF	0.986	KNN	0.984
Layer-2	MLP	0.913	MLP	0.951

Table 5. Performance of the best model for each layer and classifier of the hierarchical 3-step model

		CM-CIC-IDS2017		CM-UNSW-NB15	
		Model	**Recall**	**Model**	**Recall**
Layer-1		RF	0.986	KNN	0.984
Layer-2		DT	0.945	MLP	0.963
Layer-3	Reconnaissance	RF	0.996	KNN	0.996
	Access	RF	0.807	MLP	0.879
	DoS	KNN	0.999	MLP	0.998
	Malware	AdaBoost	0.934	MLP	0.989

Table 5 shows the best model selected for each layer and its recall performance on the training data as a result of the Optimal Model Selection experiment using GridSearchCV.

4.2.2. Hierarchical Framework Evaluation

To assess the overall performance of the entire architecture, we conducted evaluations using the same test data as utilized in section 4.2.1. The performance of our classification was evaluated primarily based on the recall metric as described in 4.1.2.

Table 6. Performance Improvement of hierarchical classifiers (2-step, 3-step) against non-hierarchical classifiers

	CM-CIC-IDS2017				CM-UNSW-NB15			
Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
1-step	0.981	0.980	**0.644**	0.981	0.988	0.986	**0.754**	0.988
2-step/3-step	0.977	0.972	**0.977**	0.973	0.988	0.988	**0.988**	0.987
Enhancement Ratio of Recall(%)	151.708				131.034			

When comparing the non-hierarchical 1-step model with the hierarchical 2-step and 3-step models, we observed a significant improvement in recall values. In other words, we noticed a notable performance difference in classifying intrusion types in detail based on the presence of Layer-1. However, distinguishing between the 2-step and 3-step models in terms of performance was challenging. We found similar performance across all metrics. This is attributed to the abundance of benign traffic compared to malicious traffic, making it difficult for performance improvements at a detailed level to be reflected in the metrics.

Detailed analysis of the confusion matrix revealed the potential for improving the accuracy of classes with high misclassification rates in the 3-step model. In the case of the DoS class, the 2-step model achieved an accuracy of 6.26%. However, the 3-step model demonstrated a substantial improvement, reaching an accuracy of 43.80%, nearly sevenfold. A similar trend was observed for the Fuzzers class, which saw its accuracy increase from 12.20% to 19.80%. Particularly noteworthy was the Shellcode class, which was entirely misclassified in the 2-step model but achieved an accuracy of around 15.50% in the 3-step model. In addition, the Web Attack-Sql Injection class saw its accuracy improve from 0.53% to 3.20%.

These results indicate that with a sufficient quantity of data for classes with high misclassification rates, the 3-step model has the potential to achieve even higher accuracy improvements.

4.3. Feature Selection Evaluation

4.3.1. Selected Feature

Feature selection was performed using three different approaches following the methodology described in Section 3.4. First, we utilized the XGBoost model-based information acquisition method to measure the importance of each feature and select the

top 50% of features with the highest importance to generate an initial feature set for the final feature set. Second, we analyzed the feature similarity between malicious traffic and removed features with high similarity from the feature data set. Thirdly, employing error analysis methods, we evaluated the impact of individual features on the misclassification rate and excluded features that had a significant negative impact. Finally, from the list of excluded features, we reintroduced those that had high importance in the first approach into the dataset, resulting in the selection of the ultimate dataset. The results for features specific to each approach are explained in Appendix C[34].

Table 7 represents the final feature set for each dataset selected through three approaches. For each dataset, 17 features were selected as final features for CM-CIC-IDS2017, and 40 features were selected as final features for CM-UNSW-NB15.

Table 7. Final selected features for each dataset

CM-CIC-IDS2017	CM-UNSW-NB15
bwd_pkts_s, flow_duration, fwd_iat_min, fwd_iat_tot, flow_iat_mean, subflow_bwd_pkts, tot_bwd_pkts, fwd_iat_mean, fwd_pkt_len_min, tot_fwd_pkts, fwd_iat_std, protocol, pkt_len_max, fwd_act_data_pkts, bwd_iat_mean, fwd_pkt_len_std, pkt_size_avg	proto_udp, dmeansz, sintpkt, sloss, dintpkt, state_RST, sbytes, synack, res_bdy_len, service_-, dwin, dttl, swin, ct_state_ttl, service_ssh, dpkts, ackdat, state_REQ, state_CON, state_FIN, service_http, sttl, service_0, dbytes, sload, proto_l2tp, proto_secure-vmtp, proto_vrrp, proto_ddx, proto_wb-mon, proto_ib, proto_trunk-2, proto_fc, proto_srp, proto_etherip, proto_mhrp, service_dhcp, proto_ip, proto_encap, proto_vines

4.3.2. Performance and Execution Time Evaluation

In our experiments, we applied the feature selection method proposed in Section 4.3.1 based on the same test data set used in Section 4.2.2. The method was applied not only to the single layer of the 1-step model, but also to the 2-step and 3-step models consisting of multiple layers. The results before applying feature selection can be found in Section 4.2.2. After application, the recall performance of the 2-step and 3-step models increased by 0.1%, especially on the CM-CIC-IDS2017 dataset. Furthermore, the performance difference depends on the number of layers: compared to the 2-step model, the 3-step model improves recall performance by an additional 0.1% on the CM-CIC-IDS2017 dataset. However, the dataset used had a bias toward certain intrusion types. Further performance improvements are expected if feature selection is applied to a dataset that reduces this bias.

Table 8. Test results of 2-step and 3-step on two CM Datasets after applying Feature Selection

	CM-CIC-IDS2017				CM-UNSW-NB15			
Classifier	Accuracy	Precision	**Recall**	F1-Score	Accuracy	Precision	**Recall**	F1-Score
2-step	0.978	0.971	**0.978**	0.973	0.988	0.985	**0.988**	0.986
3-step	0.978	0.972	**0.978**	0.973	0.988	0.986	**0.988**	0.985

Based on the results above, it is evident that the CM-UNSW-NB15 dataset is a more effective dataset for intrusion classification. Accordingly, we will use the data format of CM-UNSW-NB15 as the format for the dataset to be collected for future intrusion detection system development, with specific details to be discussed in the subsequent discussion section.

5. Discussion

In future research, it is crucial to explore diverse network environments and intrusion scenarios. We plan to collect datasets that reflect real-world intrusion trends and dynamics

to enhance the model's generalization performance. Additionally, addressing the class imbalance issue is a priority. We will develop and implement effective strategies, including data augmentation, sampling techniques, and meta-learning algorithms, to mitigate misclassification rates in classes with high imbalances. Furthermore, we will research to develop real-time network intrusion detection and response capabilities, enabling swift and effective responses when intrusions are detected.

6. Conclusions

In this paper, we propose a hierarchical intrusion detection model with two or more layers based on machine learning models, named the 2-step model and the 3-step model. The proposed model classifies intrusion types sequentially from Layer-1 to Layer-3. In addition, we introduce a new feature selection methodology, which is a dual perspective analysis based on the feature importance score calculated by the Ensemble learning. The feature selection results showed a recall value of more than 97.7% for the 3-step Model, with better performance on the CM-UNSW-NB15 dataset. Overall, our model has the highest accuracy of 98.8%, precision of 98.6%, recall of 98.8% and F1-score of 98.5%. This indicates that the proposed model performs well in detecting and classifying intrusion types and can effectively respond to intrusion detection in the network even when different countermeasures are applied to each intrusion type in the future.

Author Contributions: Conceptualization, Yunji Kim, Junghye Kim, Jihyeon Kim, and Dongho Kim; Methodology, Yunji Kim, Junghye Kim, Jihyeon Kim, and Dongho Kim; Funding acquisition, Dongho Kim; Investigation, Yunji Kim, Junghye Kim, Jihyeon Kim, and Dongho Kim; Project administration, Dongho Kim; Software, Yunji Kim and Junghye Kim; Supervision, Dongho Kim; Writing – original draft, Yunji Kim, Junghye Kim and Jihyeon Kim; Writing – review & editing, Dongho Kim; All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (No. S-2021-A0496-00167).

Institutional Review Board Statement: Not applicable

Informed Consent Statement:

Data Availability Statement: CM-CIC-IDS2017, https://github.com/CSID-DGU/MLAC/blob/main/Consolidated_data/CM-CIC-IDS2017.csv.dvc; CM-UNSW-NB15, https://github.com/CSID-DGU/MLAC/blob/main/Consolidated_data/CM-UNSW-NB15.csv.dvc;

Acknowledgments:

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rekha, G.; Malik, S.; Tyagi, A.K.; Nair, M.M. Intrusion detection in cyber security: role of machine learning and data mining in cyber security. *Advances in Science, Technology and Engineering Systems Journal* **2020**, *5*, 72–81.
2. Hofstede, R.; Čeleda, P.; Trammell, B.; Drago, I.; Sadre, R.; Sperotto, A.; Pras, A. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials* **2014**, *16*, 2037–2064.
3. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSp, 2018, pp. 108–116.
4. Moustafa, N.; Slay, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) **2015**.
5. Liao, H.J.; Richard Lin, C.H.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications* **2013**, *36*, 16–24. https://doi.org/https://doi.org/10.1016/j.jnca.2012.09.004.
6. Ring, M.; et al. A survey of network-based intrusion detection data sets. *Computers & Security* **2019**, *86*, 147–167.
7. Rosay, A.; Carlier, F.; Cheval, E.; Leroux, P. From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2021, pp. 570–575.
8. Rosay, A.; Cheval, E.; Carlier, F.; Leroux, P. Network intrusion detection: A comprehensive analysis of CIC-IDS2017. In Proceedings of the 8th International Conference on Information Systems Security and Privacy. SCITEPRESS-Science and Technology Publications, 2022, pp. 25–36.

9. Lanvin, M.; Gimenez, P.F.; Han, Y.; Majorczyk, F.; Mé, L.; Totel, E. Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In Proceedings of the International Conference on Risks and Security of Internet and Systems. Springer, 2022, pp. 18–33.
10. Bhatia, S.; et al. MSTREAM: Fast anomaly detection in multi-aspect streams. In Proceedings of the Proceedings of the Web Conference 2021, 2021.
11. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: a multitiered hybrid intrusion detection system for Internet of vehicles. *IEEE Internet of Things Journal* **2021**, *9*, 616–632.
12. Belarbi, O.; et al. An Intrusion Detection System based on Deep Belief Networks. *arXiv preprint arXiv:2207.02117* **2022**.
13. Goryunov, M.N.; Matskevich, A.G.; Rybolovlev, D.A. Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset. *Proceedings of the Institute for System Programming of the RAS* **2020**, *32*, 81–94.
14. Data, M.; Aritsugi, M. T-DFNN: An Incremental Learning Algorithm for Intrusion Detection Systems. *IEEE Access* **2021**, *9*, 154156–154171.
15. Aslansefat, K.; et al. SafeML: safety monitoring of machine learning classifiers through statistical difference measures. In Proceedings of the International Symposium on Model-Based Safety and Assessment. Springer, 2020.
16. Verkerken, M.; D'hooge, L.; Sudyana, D.; Lin, Y.D.; Wauters, T.; Volckaert, B.; De Turck, F. A Novel Multi-Stage Approach for Hierarchical Intrusion Detection. *IEEE Transactions on Network and Service Management* **2023**.
17. Grashöfer, J.; et al. Advancing Protocol Diversity in Network Security Monitoring. *arXiv preprint arXiv:2106.12454* **2021**.
18. Stiawan, D.; et al. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access* **2020**, *8*, 132911–132921.
19. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of encrypted and vpn traffic using time-related. In Proceedings of the Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016, pp. 407–414.
20. Lashkari, A.H.; et al. CICFlowMeter. https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt, 2017. Accessed: 2021-08-10.
21. Quinlan, J.R. Induction of decision trees. *Machine Learning* **1986**, *1*, 81–106.
22. Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, 5–32.
23. Bayes, T. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. R. Soc.* **1763**, *53*, 370–418.
24. Fisher, R.A. The use of multiple measurements in taxonomic problems. *Annals of eugenics* **1936**, *7*, 179–188.
25. Cox, D.R. The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society. Series B (Methodological)* **1958**, *20*, 215–242.
26. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences* **1997**, *55*, 119–139.
27. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **1967**, *13*, 21–27. https://doi.org/10.1109/TIT.1967.1053964.
28. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324. https://doi.org/10.1109/5.726791.
29. Noble, W.S. What is a support vector machine? *Nature Biotechnology* **2006**, *24*, 1565–1567.
30. Kim, Y. Comparision of Machine Learning Models (Appendix A.). Available online: https://github.com/CSID-DGU/MLAC/blob/main/docs/Appendix%20A.pdf (accessed on 2023-10-13). Accessed on 13 October 2023.
31. Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine Learning in Python. *Machine Learning in Python* **2011**.
32. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2016; KDD '16, p. 785–794. https://doi.org/10.1145/2939672.2939785.
33. Kim, Y. Training Datasets Configuration of Two MC Datasets (Appendix B.). Available online: https://github.com/CSID-DGU/MLAC/blob/main/docs/Appendix%20B.pdf (accessed on 2023-10-13). Accessed on 13 October 2023.
34. Kim, Y. Training Datasets Configuration of Two MC Datasets (Appendix B.). Available online: https://github.com/CSID-DGU/MLAC/blob/main/docs/Appendix%20C.pdf (accessed on 2023-10-13). Accessed on 13 October 2023.