IEEE Open Journal of the Communications Society

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 11 January, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

Development of Various Stacking Ensemble Based HIDS Using ADFA Datasets

HAMi SATILMIŞ[1], SEDAT AKLEYLEK[2, 3], AND ZALiHA YÜCE TOK[4]

[1]Department of Computer Engineering, Ondokuz Mayıs University, Samsun 55200, Türkiye
[2]Department of Computer Engineering, İstinye University, 34010 Istanbul, Türkiye
[3]Chair of Security and Theoretical Computer Science, University of Tartu, Tartu, Estonia
[4]ASELSAN, Ankara, Türkiye

CORRESPONDING AUTHOR: Hami Satılmış (hami.satilmis@bil.omu.edu.tr).

Zaliha Yüce Tok was supported by TUBITAK 1515 Frontier RD Laboratories Support Program.

ABSTRACT The rapid increase in the number of cyber attacks and the emergence of various attack variations pose significant threats to the security of computer systems and networks. Various intrusion detection systems (IDS) are developed to defend computer systems and networks in response to these threats. One type of IDS, known as a host-based intrusion detection system (HIDS), focuses on securing a single host. Numerous HIDS have been proposed in the literature, incorporating various detection methods. This study develops multiple machine learning (ML) models and stacking ensemble based HIDS that can be used as detection methods in HIDS. Initially, n-grams, standard bag-of-words (BoW), binary BoW, probability BoW, and term frequency-inverse document frequency (TF-IDF) BoW methods are applied to the ADFA-LD and ADFA-WD datasets. Mutual information and k-means methods are used together for feature selection on the resulting BoW datasets. Individual models are created using either selected features or all features. Subsequently, the outputs of these individual models are used in extreme gradient boosting (XGBoost) and adaptive boosting (AdaBoost) models to develop stacking ensemble based models. The experimental results show that the best accuracy (ACC) among models using ADFA-LD based BoW datasets is achieved by the stacking ensemble based XGBoost model, which has an ACC of 0.9747. This XGBoost model utilizes the standard BoW dataset and selected features. Among models using ADFA-WD based BoW datasets, the stacking ensemble based XGBoost is also the most successful in terms of ACC, with an ACC of 0.9163, using the standard BoW dataset and all features.

INDEX TERMS Intrusion Detection System, Host-based Intrusion Detection System, Information Security, Machine Learning.

I. INTRODUCTION

In today's world, technological innovations in computer systems continue unabated, and the volume of data flow between these systems is increasing daily. These developments attract the attention of malicious users or attackers and motivate them to engage in malicious activities. Attackers discover security vulnerabilities in computer systems or networks through various methods. Additionally, attackers carry out various cyberattacks to exploit their discovered security vulnerabilities. Multiple types of cyberattacks, such as malware, threaten the fundamental concepts of information security, including confidentiality, integrity, and availability. Therefore, the detection of cyberattacks is defined as a crucial action for ensuring information security.

Various solutions, such as intrusion detection systems (IDS), are proposed to ensure information security. An IDS detects various cyberattacks targeting computer systems or networks. IDSs are typically deployed on hosts or network devices and monitor activities within the systems where they are placed. In addition to the monitoring function, IDSs collect various data about the activities they observe. Subsequently, IDSs process the collected data using multiple methods [1], [2]. IDSs that analyze the data can determine whether the activities involve cyberattacks. IDSs are

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

classified into two types based on the data source they collect. These two types of IDS are network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS) [2]–[4].

HIDS monitors the activities of a host. During monitoring, HIDS collects data such as system calls, API calls, dynamic link library (DLL) files, and system logs associated with the host [5], [6]. The collected data is analyzed by HIDS using various methods to detect cyberattacks within the activities. HIDS are categorized into two types based on their detection methods: signature-based and anomaly-based [7]–[9]. Signature-based HIDS stores signatures of various attacks in their databases. These HIDS detect cyberattacks by comparing newly collected signatures with the stored ones, thus identifying known attacks. Therefore, signature-based HIDS or IDS may remain ineffective against zero-day attacks [10]. On the other hand, anomaly-based HIDS employs various statistical methods or machine learning (ML)/deep learning (DL) techniques as detectors. These methods in anomaly-based HIDS are trained on data representing normal activities, thereby creating profiles of normal behavior [5]. This HIDS identifies activities that deviate from these profiles as potential cyberattacks or malicious activities. On the other hand, most ML/DL models in HIDS are generally defined as complex due to the large number of features they involve. This complexity can lead to high variance or bias problems [11]. To mitigate these issues, feature selection methods are employed to reduce the dimensionality of features, or ensemble based models that combine various individual models are developed [11].

A. Related Studies

In [12], a new HIDS was proposed to detect malicious behaviors on Android devices. This HIDS employed various statistical methods and ML models as detectors. These detectors analyzed system log files to identify suspicious behaviors and calculated the percentage likelihood of these behaviors being malicious. The HIDS was developed and evaluated on two real-time datasets obtained from an Android device. In the experiments, the HIDS, which included the univariate Gaussian model (UGM) or the multivariate Gaussian model (MGM) as statistical methods, achieved an accuracy (ACC) of 100%. The HIDS that included ML models such as one rule (OneR), decision tree (DT), naive Bayes (NB), Bayesian network (BN), logistic regression (LR), random forest (RF), k-nearest neighbors (KNN), and support vector machine (SVM) as detectors demonstrated 100% ACC performance, except the NB model.

In [13], a HIDS was designed to detect attacks targeting internet of things (IoT) devices. In this automated HIDS, user-space and kernel-space data were utilized. These data were combined with ML/DL models. In the HIDS, system call sequences from devices were collected using the LTTng tracer [14], and features were extracted by the Babeltrace API. The extracted features were represented using a one-hot encoder and provided as input to ML/DL models. The HIDS employed models such as DT, RF, gradient boosted trees (GBT), SVM, multilayer perceptron (MLP), and one-class SVM (OCSVM) as detectors. These models were developed using a dataset containing data from a home automation system. The HIDS achieved 100% accuracy (ACC) with the GBT model regarding anomaly detection performance in the experiments conducted.

In [15], system call sequences used in ML models as detectors in HIDS were preprocessed using bag-of-word (BoW) methods. Standard BoW, boolean BoW, probability BoW, term frequencies, and inverse document frequencies (TF-IDF) BoW methods were employed in this preprocessing stage. The system call sequences that underwent this preprocessing were used in the development and evaluation of J48, RF, RIPPER, KNN, SVM, and NB models. The models were trained and evaluated on the Australian Defence Force Academy Linux dataset (ADFA-LD) [16] and the virtual machine monitor (VMM) malware dataset. In the experiments on ADFA-LD, the RF model using the standard BoW method achieved the highest performance with an ACC of 0.9840 and a false positive rate (FPR) of 0.0170. Meanwhile, among the models developed using the VMM dataset, the J48 and RF models that employed the TF-IDF BoW method demonstrated the best performance with an ACC of 1.0000 and a false alarm rate (FAR) of 0.0000.

In [17], a new method for feature selection, termed context-aware feature extraction, was introduced. This method was also designed to reduce classification time. Additionally, an IDS utilizing this feature selection method was proposed. This IDS, which employed a convolutional neural network (CNN) model as the detector, was designed to detect nearly 12 attacks. The IDS was developed and evaluated using the NSL-KDD [18], CICIDS2017 [19], ADFA-LD, and Australian Defence Force Academy Windows dataset (ADFA-WD) [20]. In experiments, the baseline CNN-based IDS demonstrated performance with ACC values of 81.90%, 99.22%, 95.12%, and 77.01% on the NSL-KDD, CICIDS2017, ADFA-LD, and ADFA-WD datasets, respectively. On the other hand, the fine-tuned CNN-based IDS achieved ACC values of 83.43%, 99.29%, and 95.34% on the NSL-KDD, CICIDS2017, and ADFA-LD datasets, respectively. In terms of classification time per sample, the baseline CNN-based IDS operated at 44.00 ms, 87.58 ms, 47.69 ms, and 43.63 ms on the NSL-KDD, CICIDS2017, ADFA-LD, and ADFA-WD datasets, respectively. The fine-tuned CNN-based IDS performed classification at 47.71 ms, 87.68 ms, and 49.72 ms per sample on the NSL-KDD, CICIDS2017, and ADFA-LD datasets, respectively.

In [21], a real-time HIDS was proposed to detect abnormal system processes. In this HIDS, system calls were represented using n-grams, and the TF-IDF values of these n-grams were calculated using the TFIDFVectorizer method. Additionally, the truncated singular value decomposition (SVD) technique was employed to reduce the dimensionality

of feature vectors associated with the n-grams. The HIDS, which included SVM, neural network (NN), and DT models as detectors, was developed and evaluated using the ADFA-LD and ADFA-WD datasets. In the experiments on ADFA-LD, the HIDS incorporating the SVM model demonstrated the best performance with FPR of 3.34% in binary classification and 9.12% in multi-class classification. In the experiments on ADFA-WD, the HIDS incorporating the NN model achieved the best results with FPRs of 8.63% in binary classification and 15.11% in multi-class classification.

In [22], a new application-level anomaly detection (ALAD) classifier was introduced for HIDSs that detects anomalies using DL methods. ALAD was designed to distinguish between normal and abnormal behaviors at the application level. The compatibility of ALAD with DL models was tested using WaveNet [23], long short-term memory (LSTM) [24], and CNN/recurrent neural networks (CNN/RNN) models [25]. ALAD, which detects abnormal behaviors, was trained and evaluated on the ADFA-LD and PLAID [22] datasets. The performance of ALAD was compared to that of another classifier, trace-level anomaly detection (TLAD). In experiments on ADFA-LD, ALAD achieved performance metrics of 99.8% area under the curve (AUC) with WaveNet, 96.6% AUC with LSTM, and 98.5% AUC with CNN/RNN. On the PLAID dataset, ALAD reached AUC values of 99.3% with WaveNet, 99.5% with LSTM, and 99.3% with CNN/RNN. Additionally, ALAD outperformed TLAD on both datasets.

In [5], a new HIDS based on stacking ensemble methods was introduced. In this HIDS, detectors included LSTM, bidirectional LSTM (Bi-LSTM), gated recurrent unit (GRU), and bidirectional GRU (Bi-GRU) models. The models in the HIDS were trained and evaluated using the ADFA-WD dataset. Before training the LSTM and GRU models, the DLL call sequences were processed using n-gram, Word2Vec, and GloVe methods. The outputs from the Word2Vec embedding method were used as inputs to the LSTM models, while those from the GloVe embedding method were used as inputs to the GRU models. Additionally, the outputs from the LSTM and GRU models were combined using a fully connected neural network (FCNN), which served as the final decision-making model. In the binary classification experiments, the HIDS demonstrated performance with an ACC of 0.9110 and an FPR of 0.1330.

In [26], a HIDS named self-evolving HIDS (SEHIDS) was proposed for detecting attacks in IoT networks. In SEHIDS, It was utilized self-evolving and architecture-updating artificial neural network (ANN) models. It was designed so that ANN models detected attacks independently on each IoT node. The SEHIDS was categorized into two types: signature-based and anomaly-based. An MLP model was used in the signature-based SEHIDS. In the anomaly-based SEHIDS, attacks were detected using a Replicator Neural Network (ReNN) model. The SEHIDS was developed and evaluated using the BoT-IoT [27], TON-IoT [28], and IoTID20 [29] datasets. In the experiments on the three datasets, the signature-based SEHIDS demonstrated performance with the lowest true positive rate (TPR) of 99% and the highest TPR of 100%. The anomaly-based SEHIDS achieved TPR values ranging from 99.9% to 100%.

In [30], a new attack detection framework was proposed that detected attacks by analyzing system call sequences. This framework utilized a hybrid model incorporating an LSTM model with a system call frequency-based anomaly detection technique. The LSTM detected attacks using data from previously analyzed system call sequences. The frequency-based technique, which employed BoW, n-gram, and TF-IDF methods during the preprocessing phase, classified the types of attacks. A weighted average ensemble model precisely predicted the classes of system call sequences. The framework was developed and evaluated using the ADFA-LD dataset. The framework achieved performance metrics of 0.9720 ACC and 0.0240 FPR in the experiments.

B. Motivation and Contribution

The literature proposes various detection and preprocessing methods that can be used within HIDSs. Table 1 includes some detection and preprocessing methods used on the ADFA-LD and ADFA-WD datasets. On the other hand, when examining the best experimental results in Table 1, it is observed that high ACC values are generally achieved while detecting attacks or anomaly events. However, detecting and binary classifying attacks or anomaly events on the ADFA datasets with higher or similar ACC values using different detection and preprocessing methods remains a primary aim. In addition to achieving high ACC rates, achieving low detection or classification times is also an objective. In this study, the primary objective is defined as the development of various ML models and stacking ensemble based HIDS that demonstrate high ACC and low FPR values. Additionally, it is aimed to develop different versions of the ML models and HIDS with low detection and classification times without significantly affecting their ACC and FPR values. Within the scope of these aims, the contributions of this study are listed below.

- In addition to the BoW methods in [15], n-grams are used to process the ADFA-LD and ADFA-WD datasets. As a result, eight datasets based on ADFA-LD and ADFA-WD are obtained, with features consisting of n-grams, including standard BoW, binary BoW, probability BoW, and TF-IDF BoW.
- To reduce the dimensions and select optimal features for the BoW datasets based on ADFA-LD and ADFA-WD, a feature selection method that combines the mutual information (MI) method with the k-means clustering algorithm is employed. As a result, the detection and classification times of ML models are reduced with BoW datasets with reduced dimensions and containing optimum features.

TABLE 1. Methods and experimental results in related studies

Study	Used Detection Methods	Used Preprocessing Methods	Best Performance Values
[15]	J48, RIPPER, RF, SVM, NB, and KNN.	Standard BoW. Boolean BoW. Probability BoW. TF-IDF BoW.	On ADFA-LD, RF with 98.4% ACC and 1.7% FPR.
[17]	CNN.	Extremely randomized trees (ERT). Select k-best (SKB).	95.34% ACC on ADFA-LD. 77.01% ACC on ADFA-WD.
[21]	SVM, NN, and DT.	n-grams. TFIDFVectorizer. SVD.	SVM with 3.34% FPR in binary classification on ADFA-LD. SVM with 9.12% FPR in multi-class classification on ADFA-LD. NN with 8.63% FPR in binary classification on ADFA-WD. NN with 15.11% FPR in multi-class classification on ADFA-WD.
[22]	WaveNet. LSTM. CNN/RNN. ALAD and TLAD.	No information.	99.8% AUC on ADFA-LD with ALAD and WaveNet combination.
[5]	LSTM. Bi-LSTM. GRU. Bi-GRU. FCNN.	n-grams. Word2Vec. Glove.	In binary classification on ADFA-WD, 91.1% ACC. In multi-classification on ADFA-WD, 68.7% ACC.
[30]	LSTM. NN+RF.	BoW. n-grams. TF-IDF.	97.2% ACC on ADFA-LD.
This Study	KNN. DT. LR. RF. XGBoost. AdaBoost.	n-grams. Standard BoW. Binary BoW. Probability BoW. TF-IDF BoW. MI+k-means.	On ADFA-LD, the stacking ensemble model XGBoost with 97.47% ACC. On ADFA-WD, the stacking ensemble model XGBoost with 91.63% ACC.

- Various individual and stacking ensemble based ML models are developed using BoW datasets based on ADFA-LD and ADFA-WD. More specifically, stacking ensemble based extreme gradient boosting (XGBoost) and adaptive boosting (AdaBoost) models are created by combining the outputs of individual models like KNN, DT, LR, and RF. In this way, different stacking ensemble based HIDS are obtained.
- The performances of the most successful models developed with BoW datasets based on ADFA-LD are compared with the performance of the RF model in [15] and the framework in [30].
- The models developed using BoW datasets based on ADFA-WD that achieve the best performance values are compared with the HIDS in [5].

C. Organization

In this study, Section II provides information about the datasets used in developing and evaluating the models. Section III discusses various methods used in the preprocessing phase and feature selection during model development. Section IV describes the development processes of the models and presents a comparative analysis of the models' performance values. Section V provides a general discussion of the developed models, methods used, and HIDSs. The final section provides a brief overview of this study. The abbreviations and their expansions used in this study are provided in Table 2.

II. Datasets

This section discusses the ADFA-LD and ADFA-WD datasets used in the development and evaluation stages of the models.

TABLE 2. List of abbreviations

IDS	Intrusion Detection System	**MI**	Mutual Information
NIDS	Network-based Intrusion Detection System	**LSTM**	Long Short Term Memory
HIDS	Host-based Intrusion Detection System	**TF-IDF**	Term Frequencies and Inverse Document Frequencies
ML	Machine Learning	**NN**	Neural Network
DL	Deep Learning	**GRU**	Gated Recurrent Unit
BoW	Bag-of-Word	**MLP**	Multilayer Perceptron
DLL	Dynamic Link Library	**F1**	F1-score
ACC	Accuracy	**TP**	True Positive
DT	Decision Tree	**FN**	False Negative
SVM	Support Vector Machine	**PRC**	Precision
NB	Naive Bayes	**REC**	Recall
CNN	Convolutional Neural Network	**FPR**	False Positive Rate
LR	Logistic Regression	**RNN**	Recurrent Neural Networks
RF	Random Forest	**IoT**	Internet of Things
XGBoost	Extreme Gradient Boosting	**AdaBoost**	Adaptive Boosting

A. ADFA-LD

The ADFA-LD dataset contains system call sequences varying in length from the Linux operating system. Each system call sequence consists of unique IDs representing system calls. Figure 1 shows an example of a system call sequence from this dataset. ADFA-LD is divided into three subsets: training, validation, and attack. The training and validation subsets contain normal-type system call sequences. The attack subset includes system call sequences related to six different attack types. The numbers of system call sequences and attack types in ADFA-LD are given in Table 3.

TABLE 3. Numbers of system call sequences and attack types in ADFA-LD

Training	Validation	Attack	Total
		Add Superuser: 91	
		FTP password bruteforce: 162	
		SSH password Bruteforce: 176	
833	4372	Java Meterpreter: 124	5951
		Linux Meterpreter: 75	
		Web shell attack: 118	
		Total: 746	

The number of normal-type system call sequences in ADFA-LD is approximately seven times greater than that of attack-type sequences. Thus, it is concluded that there is an imbalance in this dataset. Considering this, this study aims to use a balanced dataset containing system call sequences from ADFA-LD. To this end, the training and attack subsets of the ADFA-LD dataset are combined. Consequently, a balanced dataset is created, consisting of 833 normal and 746 attack system call sequences, totaling 1579 sequences. This balanced dataset is used in the development and evaluation phases of the models.

B. ADFA-WD

The ADFA-WD dataset consists of DLL call sequences from the Windows XP operating system. Each DLL call sequence is represented by DLL calls identified by DLL names and specific memory access addresses. An example of a DLL call sequence from this dataset is shown in Figure 2. ADFA-WD includes three subsets: training, validation, and attack. The training and validation subsets contain normal-type DLL call sequences, with 355 and 1827 sequences, respectively. The attack dataset includes 5542 DLL call sequences covering 12 different attack types. The numbers of DLL call sequences and attack types in ADFA-WD are given in Table 4.

TABLE 4. Numbers of system call sequences and attack types in ADFA-WD

Training	Validation	Attack	Total
		V1-CesarFTP : 454	
		V2-WebDAV : 470	
		V3-Icecast : 382	
		V4-Tomcat : 418	
		V5-OS-SMB: 355	
		V6-OS-Print-Spool: 454	
355	1827	V7-PMWiki: 430	7724
		V8-Wireless-Karma: 487	
		V9-PDF: 440	
		V10-Backdoored Executable: 536	
		V11-Browser-Attack: 495	
		V12-Infectious-Media: 621	
		Total: 5542	

The number of attack-type DLL call sequences in ADFA-WD is approximately 2.5 times greater than that of normal-type sequences. Thus, there is an imbalance problem in this dataset. In response to this issue, this study aims to use a balanced dataset consisting of DLL call sequences from ADFA-WD. To achieve this goal, the training and validation subsets of the dataset are combined. As a result, 2182 normal-type DLL call sequences are gathered. The attack-type DLL call sequences are selected to balance with the number of normal-type sequences. In the selection process, 40% of the DLL call sequences in the attack subset are randomly chosen.

```
[6, 11, 45, 33, 192, 33, 5, 197, 192, 6, 33, 5, 3, 197, 192, 192, 6, 33
, 5, 3, 197, 192, 192, 6, 33, 5, 3, 197, 192, 192, 192, 6, 33, 5, 3, 19
7, 192, 192, 6, 33, 5, 3, 197, 192, 192, 192, 6, 33, 5, 3, 197, 192, 19
2, 6, 33, 5, 3, 197, 192, 192, 192, 6, 192, 192, 243, 125, 125, 125, 12
5, 125, 125, 125, 125, 125, 91, 258, 311, 240, 240, 174, 174, 175, 191,
122, 268, 45, 45, 5, 197, 192, 3, 3, 6, 91, 5, 197, 192, 3, 3, 6, 91, 2
01, 54, 195, 196, 196, 38, 6, 6, 6]
```

FIGURE 1. A system call sequence in the ADFA-LD dataset

```
['ntdll.dll+0x16d33', 'ntdll.dll+0x16f03', 'ntdll.dll+0x1ce16',
'ntdll.dll+0x1ccd2', 'ntdll.dll+0x16071', 'ntdll.dll+0x162da',
'kernel32.dll+0x1bb9', 'kernel32.dll+0xace4', 'ntdll.dll+0x16d33',
'ntdll.dll+0x16f03', 'ntdll.dll+0x1ce16', 'ntdll.dll+0x1ccd2',
'ntdll.dll+0x16071', 'ntdll.dll+0x162da', 'kernel32.dll+0x1bb9',
'kernel32.dll+0xace4', 'ntdll.dll+0x1cd1b', 'ntdll.dll+0x16071',
'ntdll.dll+0x162da', 'kernel32.dll+0x1bb9', 'kernel32.dll+0xace4',
'ntdll.dll+0x1cd1b', 'ntdll.dll+0x16071', 'ntdll.dll+0x162da',
'kernel32.dll+0x1bb9', 'kernel32.dll+0xace4', 'ntdll.dll+0x1cd1b',
'ntdll.dll+0x16071', 'ntdll.dll+0x162da', 'kernel32.dll+0x1bb9']
```

FIGURE 2. A DLL call sequence in the ADFA-WD dataset

Thus, 2217 attack-type DLL call sequences are obtained. Consequently, a balanced dataset consists of 2182 normal and 2217 attack-type DLL call sequences, totaling 4399. This balanced dataset is used to develop models and measure their performance.

III. Preprocessing and Feature Selection

In this section, the application of n-gram and BoW methods on the ADFA-LD and ADFA-WD datasets, as well as the process of creating BoW datasets, is discussed. Additionally, the procedure for reducing the dimensions of the created BoW datasets and selecting optimal features is explained.

A. Application of N-Gram Method

To capture the relationships and transitions between system/DLL calls, n-grams of various lengths are generated from raw call sequences. In finding n-grams within call sequences, a window size is determined based on the value of n. Subsequently, the window is positioned to encompass the first system/DLL call in the sequence, identifying the first n-gram. To identify subsequent n-grams, the window is shifted to the right, with each shift starting from the next system/DLL call. This process continues until the window reaches the last system/DLL call in the sequence, thereby identifying all n-grams within the call sequence.

For example, Figure 3 illustrates the process of identifying n-grams for two system call sequences of varying lengths, similar to those found in ADFA-LD. Figure 4 displays a small portion of the n-grams and their counts obtained after applying the n-gram identification process to system call sequences in ADFA-LD. Additionally, Figure 5 presents the n-grams obtained after applying the process outlined in Figure 3 to three DLL call sequences of different lengths, similar to those in ADFA-WD.

In the operation conducted on the ADFA datasets with an n value of 5, 80,526 unique 5-grams are obtained from the system call sequences in ADFA-LD. Similarly, 40,752 unique 5-grams are identified from the 4,399 DLL call sequences in ADFA-WD.

After identifying the 5-grams from the call sequences in the datasets, each 5-gram is assigned a numerical label. For example, ”N-gram_1” represents the first 5-gram. Following the labeling process, 5-gram datasets are created, where each 5-gram is treated as a feature. In these 5-gram datasets, the value of each 5-gram for a particular call sequence corresponds to its frequency within that sequence. As a result, two 5-gram datasets corresponding to the ADFA datasets are obtained. An example of a 5-gram dataset, constructed from the n-grams shown in Figure 3, is presented in Figure 6.

B. Application of BoW Methods

Standard BoW, binary BoW, probability BoW, and TF-IDF BoW methods, which are frequently used in the fields of text mining and natural language processing, are applied to the 5-gram datasets derived from the ADFA datasets. During the application process, the new values of the 5-grams, which are the features of the datasets, are calculated by the BoW methods. As a result of the application, BoW datasets corresponding to each BoW method are created. These BoW datasets are used in ML models' training and testing phases.

1) Standart BoW

The standard BoW method calculates the frequency of words within text datasets. Initially, it identifies the unique words in the texts. Each unique word is considered a feature. The values of these features correspond to the frequency of the

This article has been accepted for publication in IEEE Open Journal of the Communications Society. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2025.3538101

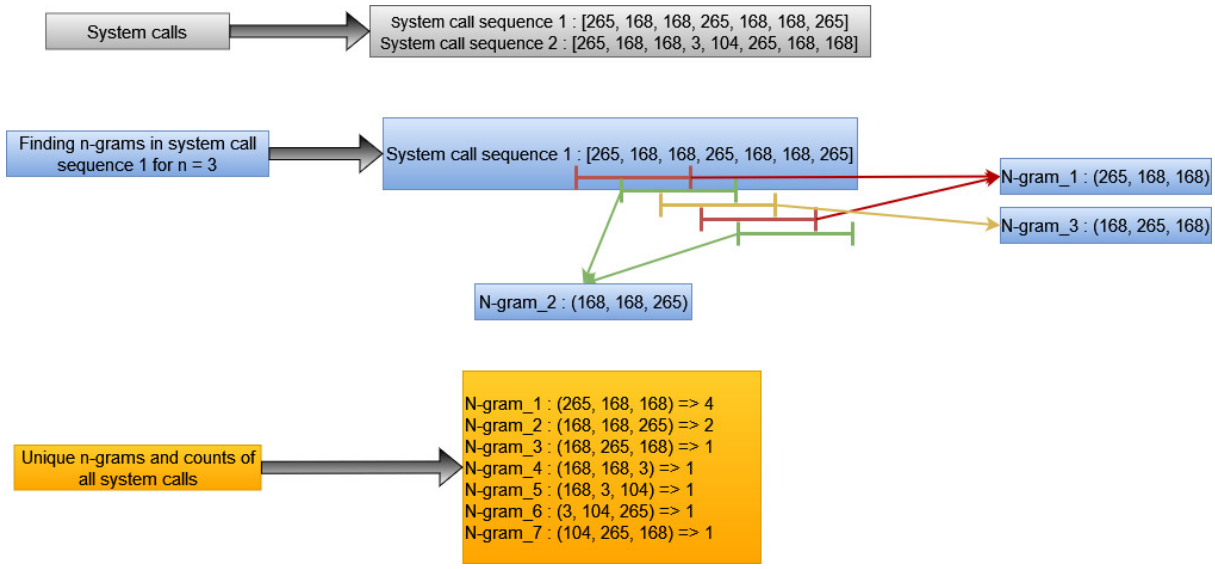

FIGURE 3. **The process of finding unique n-grams of call sequences**

```
(6, 6, 63, 6, 42): 1
(6, 63, 6, 42, 120): 10
(63, 6, 42, 120, 6): 16
(6, 42, 120, 6, 195): 4
(42, 120, 6, 195, 120): 2
(120, 6, 195, 120, 6): 2
(6, 195, 120, 6, 6): 1
(195, 120, 6, 6, 114): 4
(120, 6, 6, 114, 114): 15
(6, 6, 114, 114, 1): 1
(6, 114, 114, 1, 1): 1
(114, 114, 1, 1, 252): 1
(114, 1, 1, 252, 252): 1
(1, 1, 252, 252, 252): 18
(1, 252, 252, 252, 1): 9
(252, 252, 252, 1, 1): 18
(252, 252, 1, 1, 1): 15
(252, 1, 1, 1, 1): 28
```

FIGURE 4. **A small part of unique n-grams and their counts of system call sequences in ADFA-LD**

words within the text. Thus, texts are represented by the unique words they contain and the frequency values of those words.

When the standard BoW method is applied to 5-gram datasets, each 5-gram is treated as a word. The value of each 5-gram is calculated similarly to the frequency of unique words within a text. In the 5-gram datasets, the repetition counts of 5-grams within the call sequences are recorded. These repetition counts represent the frequency values of the 5-grams within the call sequences. Therefore, the 5-gram datasets serve as the standard BoW method applied to them. In other words, the 5-gram datasets derived from the ADFA datasets are identical to the standard BoW datasets.

2) Binary BoW

The binary BoW method focuses on whether words are present in the texts within a text dataset rather than calculating the frequency of the words. If a word, considered a feature representing a text, is found in the text, its value is set to 1. Otherwise, its value is set to 0. As a result, texts are represented by 0s and 1s corresponding to the presence or absence of words.

When the binary BoW method is applied to 5-gram datasets, the values of each 5-gram, which are treated as words, are examined. If the value of a 5-gram in a system/DLL call is greater than 0, its new value is set to 1. Conversely, if the 5-gram value is 0, it remains 0. As a result, two binary BoW datasets are created, where the 5-gram values are either 0 or 1. An example of a binary BoW dataset, obtained by applying the binary BoW method to the n-gram dataset shown in Figure 6, is illustrated in Figure 7.

3) Probability BoW

The probability BoW method calculates the likelihood of words appearing in texts. The probability value of a word is determined by dividing the frequency of the word in the text by the sum of the frequencies of all words. Thus, the texts are represented by the probability values of the words used as features.

When the probability BoW method is applied to 5-gram datasets, the probability values of the 5-grams, which are treated as words, are calculated using Equation 1. In this equation, the probability value of any 5-gram in a call sequence is determined by dividing the value of the respective 5-gram by the total value of all n-grams. After calculating the probability values of the 5-grams for each system/DLL call sequence, two probability BoW datasets containing the probability values of the 5-grams from the ADFA datasets

DLL call sequence 1 : ['ntdll.dll+0x2173e', 'kernel32.dll+0xb50b', 'kernel32.dll+0xb50b', "ntdlldll+0x2173e', 'kernel32.dll+0xb50b', 'kernel32.dll+0xb50b']
DLL call sequence 2 : ['ntdll.dll+0x2173e', 'kernel32.dll+0xb50b', 'kernel32.dll+0xb50b', 'user32.dll+0x127be', 'ws2_32.dll+0x7bf']
DLL call sequence 3 : ['kernel32.dll+Oxb50b', 'user32.dll+0x127be', 'ws2_32.dll+0x7bf', 'ntdll.dll+0x2173e', 'kernel32.dll+0xb50b', 'kernel32.dll+0xb50b', 'user32.dll+0x127be']

n = 3

N_gram_1 = ('ntdll.dll+0x2173e', 'kernel32.dll+0xb50b", 'kernel32.dll+0xb50b')
N_gram_2 = ('kernel32.dll+0xb50b', 'kernel32.dll+0xb50b', 'ntdll.dll+0x2173e')
N_gram_3 = ('kernel32.dll+0xb50b', 'ntdll.dll+0x2173e', 'kernel32.dll+0xb50b')
N_gram_4 = ('kernel32.dll+0xb50b', 'kernel32.dll+0xb50b', 'user32.dll+0x127be')
N_gram_5 = ('kernel32.dll+0xb50b', 'user32.dll+0x127be', 'ws2_32.dll+0x7b9f')
N_gram_6 = ('user32.dll+0x127be', 'ws2_32.dll+0x7b9f, 'ntdll.dll+0x2173e')
N_gram_7 = ('ws2_32.dll+0x7b9f', 'ntdll.dll+0x2173e', 'kernel32.dll+0xb50b')

FIGURE 5. **Unique n-grams of call sequences similar to DLL call sequences in ADFA-WD**

Call No	N-gram_1	N-gram_2	N-gram_3	N-gram_4	N-gram_5	N-gram_6	N-gram_7
1	2	2	1	0	0	0	0
2	2	0	0	1	1	1	1

FIGURE 6. **An example n-gram dataset**

Call No	N-gram_1	N-gram_2	N-gram_3	N-gram_4	N-gram_5	N-gram_6	N-gram_7
1	1	1	1	0	0	0	0
2	1	0	0	1	1	1	1

FIGURE 7. **An example binary BoW dataset**

are obtained. Figure 8 presents an example of a probability BoW dataset created by applying the probability BoW method to the n-gram dataset shown in Figure 6.

$$P(5-gram_i) = \frac{\text{The value of } 5-gram_i \text{ in a sequence}}{\text{Total value of } 5-grams \text{ in a sequence}} \quad (1)$$

4) TF-IDF BoW

The TF-IDF BoW method is used to determine the importance of a word within a document. The importance of a word is based on its frequency within the document and its prevalence across all documents. The method first identifies the unique words within the documents and calculates their term frequency (TF) values. Typically, the TF value of a word is its frequency within the document. The words' inverse document frequency (IDF) values measure their prevalence across all documents. The IDF value of a word is calculated by dividing the total number of documents by the number of documents containing the word and then taking the logarithm of this ratio (base 10). The product of the TF and IDF values provides the TF-IDF values of the words. Thus, documents are represented by the TF-IDF values of their unique words, each considered a feature.

Before applying the TF-IDF BoW method to the 5-gram datasets, the sequences represented by 5-grams are converted into text format. During this conversion, the labels of the 5-grams with non-zero values are used. The labels of these 5-grams are concatenated with spaces in between, according to the values of the 5-grams. Thus, each sequence is transformed into a text composed of the labels of the 5-grams. For example, the text representation of the first system call sequence from the n-gram dataset shown in Figure 6 can be found in Figure 9.

After converting each sequence in the 5-gram datasets into text format, the texts are processed using the TF-IDF BoW method. The TF-IDF values of the words (5-grams) in the texts are calculated using the TfidfVectorizer method from the scikit-learn library. The default parameters of the TfidfVectorizer method are used during this calculation. As a result, two TF-IDF BoW datasets are obtained, where the features are the TF-IDF values of the 5-grams, and these features represent the sequences. For example, the TF-IDF BoW dataset obtained from the n-gram dataset shown in Figure 6 is illustrated in Figure 10.

This article has been accepted for publication in IEEE Open Journal of the Communications Society. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2025.3538101

Call No	N-gram_1	N-gram_2	N-gram_3	N-gram_4	N-gram_5	N-gram_6	N-gram_7
1	0.400	0.400	0.200	0	0	0	0
2	0.333	0	0	0.167	0.167	0.167	0.167

FIGURE 8. An example probability BoW dataset

Text of system call sequence 1 =>	"N-gram_1 N-gram_1 N-gram_2 N-gram_2 N-gram_3"

FIGURE 9. Text of a sample system call sequence

Call No	N-gram_1	N-gram_2	N-gram_3	N-gram_4	N-gram_5	N-gram_6	N-gram_7
1	0.537	0.755	0.378	0	0	0	0
2	0.580	0	0	0.407	0.407	0.407	0.407

FIGURE 10. An example TF-IDF BoW dataset

C. Feature Selection

To reduce detection and classification times in models using BoW datasets, it is necessary to reduce the dimensionality of these datasets and select optimal features. The steps outlined in Algorithm 1 are applied to the BoW datasets to achieve this. In this algorithm, the MI method and the k-means clustering algorithm are used together.

According to Algorithm 1, the first step involves calculating the MI values of the features in the datasets. Subsequently, features are clustered into two groups based on their MI values using the k-means algorithm. One group contains features with low MI values, while the other comprises features with high MI values. Only the features from the cluster with high MI values are utilized during model development and evaluation. The statistical information regarding the features obtained and used after the feature selection process is provided in Tables 5 and 6.

Algorithm 1 Feature selection algorithm

Input: BoW dataset

Output: BoW dataset with selected features and reduced dimensionality

1: k = 2 ▷ k is the cluster number
2: Calculating MI values of features
3: k-means(k, MI values along with the indexes of the features)
4: Selecting features in the set containing large MI values according to their indexes
5: **return** BoW dataset with selected features and reduced dimensionality

IV. Development and Evaluation of Proposed Stacking Ensemble Based HIDSs

In analyzing the ADFA datasets, n-grams are extracted using an n value of 5, creating 5-gram datasets where these n-grams are considered features. These datasets are then processed using various BoW methods, each producing distinct BoW datasets. Next, feature selection is performed on the BoW datasets according to Algorithm 1. The selected features are then used to develop and evaluate the models.

For model training, 75% of the data from the BoW datasets is utilized, while the remaining 25% is used for performance evaluation. The process of developing models using BoW datasets derived from 5-gram datasets is summarized in Figure 11. Individual models are developed, including KNN, DT, LR, and RF. Following this, ensemble models such as XGBoost and AdaBoost, which combine the outputs of the individual models, are created, resulting in stacking ensemble models. As a result, stacking ensemble based HIDSs are being developed.

Additionally, the models in this study are developed using the Google Colaboratory environment with Python programming, utilizing the scikit-learn, LightGBM, and NumPy libraries.

A. Evaluation Metrics

The performance of the models developed using BoW datasets is evaluated based on several metrics, including precision (PRC), recall (REC), f1-score (F1), FPR, ACC, training time, and testing time. To calculate PRC, REC, F1, FPR, and ACC metrics, the values of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) are required. The definitions of TP, TN, FP, and FN are listed below:

- ***TP:*** The number of positive samples correctly classified as positive.
- ***TN:*** The number of negative samples correctly classified as negative.
- ***FP:*** The number of negative samples incorrectly classified as positive.

TABLE 5. Number of selected features after applying feature selection process on BoW datasets derived ADFA-LD

	Standard BoW Dataset	Binary BoW Dataset	Probability BoW Dataset	TF-IDF BoW Dataset
Raw Feature Count	80526	80526	80526	80526
Selected Feature Count	18722	18508	18634	18338

TABLE 6. Number of selected features after applying feature selection process on BoW datasets derived ADFA-WD

	Standard BoW Dataset	Binary BoW Dataset	Probability BoW Dataset	TF-IDF BoW Dataset
Raw Feature Count	40752	40752	40752	40752
Selected Feature Count	5887	589	462	461

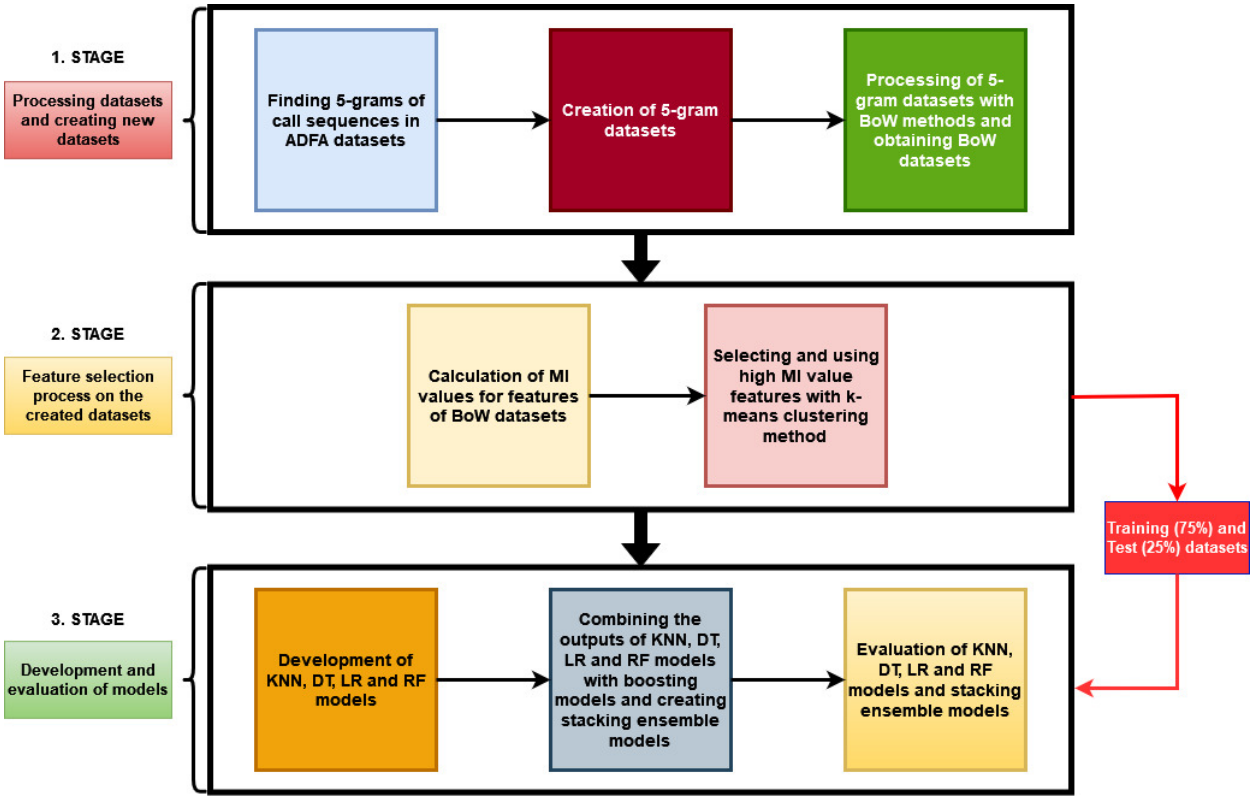

FIGURE 11. Development process of stacking ensemble models based on ADFA datasets

- ***FN:*** The number of positive samples incorrectly classified as negative.

The descriptions and mathematical equations for the metrics PRC, REC, F1, FPR, ACC, training time, and testing time are provided below.

- ***PRC:*** The ratio of the number of correctly classified positive samples to the total number of samples classified as positive.

$$PRC = \frac{TP}{TP + FP} \tag{2}$$

- ***REC:*** The ratio of the number of correctly classified positive samples to the total number of actual positive samples.

$$REC = \frac{TP}{TP + FN} \tag{3}$$

- ***F1:*** The harmonic mean of the PRC and REC values.

$$F1 = 2 * \frac{PRC * REC}{PRC + REC} \tag{4}$$

- ***FPR:*** The ratio of misclassified negative samples to total negative samples.

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

- ***ACC:*** The ratio of correctly classified samples to the total number of samples.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

- ***Training Time:*** The total time elapsed during the training of a model.
- ***Testing Time:*** The total time elapsed for a model to classify all test samples.

This article has been accepted for publication in IEEE Open Journal of the Communications Society. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2025.3538101

B. Evaluation of ADFA-LD Based Models

Tables 7 and 8 present the performance values of models using all features or selected features from ADFA-LD based BoW datasets in detecting attacks and anomaly events and in binary classification. When the models that use all the features are evaluated,

- Regarding the ACC metric, the most successful was the ensemble model XGBoost, which achieved an ACC of 0.9722. This performance value was obtained on both the standard BoW and probability BoW datasets.
- In terms of the FPR metric, the most successful model was the ensemble model AdaBoost, which achieved an FPR of 0.0236. This performance value was attained on the TF-IDF BoW dataset.
- Regarding the testing time metric, the most successful model was the DT model, which achieved a time of 0.0490 seconds. This time was obtained on the binary BoW dataset.

When examining the models developed with the selected features from the ADFA-LD based BoW datasets,

- In terms of the ACC metric, the most successful model was the XGBoost ensemble model, with an ACC of 0.9747. This performance value was achieved on the standard BoW dataset.
- Regarding the FPR metric, the most successful were the ensemble models XGBoost and AdaBoost, which both achieved an FPR of 0.0283. The XGBoost model reached this performance value on the probability BoW dataset. Meanwhile, the AdaBoost model achieved the best FPR values on both the standard BoW and probability BoW datasets.
- In terms of the testing time metric, the most successful model was LR, achieving a time of 0.0139 seconds. The LR model obtained this duration on the TF-IDF BoW dataset.

When comparing models developed using all or selected features from ADFA-LD based BoW datasets,

- In experiments conducted on the standard BoW dataset, models using all features generally exhibited better performance values regarding ACC and FPR metrics than their counterparts using selected features. The model that achieved the highest performance in terms of ACC was the XGBoost model with selected features. On the other hand, the best FPR values were reached by the XGBoost model with all features and the AdaBoost models with all or selected features.
- In experiments performed on the binary BoW dataset, models using selected features generally performed better regarding ACC and FPR metrics than their counterparts using all features. The XGBoost model with selected features achieved the highest ACC value. At the same time, the best performance in terms of FPR was demonstrated by the AdaBoost model with selected features.
- In experiments completed on the probability BoW dataset, models using selected features generally achieved better ACC and FPR metrics results than their counterparts using all features. The most successful model in terms of ACC was XGBoost, whether it used all or selected features. On the other hand, the best value for the FPR metric was achieved by the XGBoost models with either all or selected features, the AdaBoost model with selected features, and the LR model with all features.
- In experiments conducted on the TF-IDF BoW dataset, models using selected features generally performed better in terms of ACC and worse in terms of FPR compared to their counterparts using all features. The highest ACC values were achieved by the XGBoost model using either all or selected features and the AdaBoost model with selected features. In addition, the best FPR value was obtained by the AdaBoost model using all features.
- In all ADFA-LD based BoW datasets, models using selected features exhibited significantly better testing time values than their counterparts using all features.

C. Evaluation of ADFA-WD Based Models

Tables 9 and 10 contain the experimental results of models that use either all features or selected features from the ADFA-WD based BoW datasets in detecting attacks and anomaly events and in binary classification. When the models that use all the features are evaluated,

- The most successful model in terms of the ACC metric was the stacking ensemble model XGBoost, with an ACC of 0.9163. The XGBoost model achieved this performance value using the standard BoW dataset.
- The most successful model in terms of the FPR metric was the stacking ensemble model AdaBoost, with an FPR of 0.1226. The AdaBoost model achieved this performance value using the standard BoW dataset.
- LR was the most successful model in the testing time metric, with a testing time of 0.0670 seconds. The LR model was achieved using the binary BoW dataset.

When examining the models developed with the selected features from the ADFA-WD based BoW datasets,

- Regarding the ACC metric, the most successful models were the ensemble models XGBoost and AdaBoost, achieving an ACC of 0.9109. These performance values were attained using the standard BoW dataset.
- Concerning the FPR metric, the most successful models were the ensemble models XGBoost and AdaBoost, achieving an FPR of 0.1226. These values were obtained with the standard BoW dataset.

TABLE 7. Results of experiments performed with all features of ADFA-LD based BoW datasets

Standard BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.6462	0.9781	0.7783	0.4623	0.7418	1.4565	4.8532
DT	0.9211	0.9563	0.9383	0.0708	0.9418	5.8913	0.0682
LR	0.9521	0.9781	0.9649	0.0425	0.9671	58.6087	0.1174
RF	0.9323	0.9781	0.9547	0.0613	0.9569	10.0989	0.0928
XGBoost	0.9674	0.9727	0.9700	0.0283	0.9722	73.5626	5.5904
AdaBoost	0.9674	0.9727	0.9700	0.0283	0.9722	73.7034	4.6601
Binary BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.5773	1.0000	0.7320	0.6321	0.6608	0.9089	2.8401
DT	0.9211	0.9563	0.9383	0.0708	0.9418	6.9260	0.0490
LR	0.9137	0.9836	0.9474	0.0802	0.9494	47.3249	0.0769
RF	0.9184	0.9836	0.9499	0.0755	0.9519	13.8948	0.2734
XGBoost	0.9231	0.9836	0.9524	0.0708	0.9544	58.2466	3.1184
AdaBoost	0.9231	0.9836	0.9524	0.0708	0.9544	57.4651	4.4548
Probability BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8876	0.4317	0.5809	0.0472	0.7114	0.0770	3.0732
DT	0.8995	0.9289	0.9139	0.0896	0.9189	25.2534	0.0586
LR	0.9670	0.9617	0.9644	0.0283	0.9671	37.4499	0.0522
RF	0.9275	0.9781	0.9521	0.0660	0.9544	15.0325	0.0993
XGBoost	0.9674	0.9727	0.9700	0.0283	0.9722	67.1284	3.7647
AdaBoost	0.9558	0.9454	0.9505	0.0377	0.9544	67.5366	3.1948
TF-IDF BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8889	0.4372	0.5861	0.0472	0.7139	0.0859	4.0084
DT	0.8814	0.9344	0.9072	0.1085	0.9114	21.5065	0.0806
LR	0.9663	0.9399	0.9529	0.0283	0.9569	53.7843	0.1417
RF	0.9188	0.9891	0.9526	0.0755	0.9544	21.8684	0.1297
XGBoost	0.9617	0.9617	0.9617	0.0330	0.9646	86.5145	4.1783
AdaBoost	0.9714	0.9289	0.9497	0.0236	0.9544	80.0019	4.4509

- The best-performing model for the testing time metric was DT, with a time of 0.0015 seconds. This time was achieved using the TF-IDF BoW dataset.

When examining the models developed using all or selected features of the ADFA-WD based BoW datasets,

- Models that used all features generally performed better in terms of ACC and FPR metrics than their counterparts that used selected features across all datasets. Among all datasets, the most successful models in terms of ACC metrics were those XGBoost models that included all dataset features. On the other hand, regarding the FPR metric, the best value on the standard BoW dataset was achieved by XGBoost models with selected features and AdaBoost models with either all or selected features. For the Binary BoW and Probability BoW datasets, the best FPR values were obtained by XGBoost models using all of these datasets' features. Finally, for the TF-IDF BoW dataset, the AdaBoost model that included all features was the most successful regarding the FPR metric.
- In all datasets, models that used selected features had significantly better testing time values compared to their counterparts that used all features.

D. Comparison with Studies in the Literature

Table 11 contains the ACC and FPR performance values for models or frameworks based on the ADFA-LD dataset. Upon examining Table 11,

- It can be observed that the most successful model in terms of ACC and FPR metrics is the RF model in [15]. Although HIDSs in this study achieved lower accuracy compared to the RF in [15], it is essential to note that XGBoost with selected features prioritized computational efficiency and flexibility. This might account for the slight reduction in accuracy and FPR, as this HIDS was optimized for faster processing times, especially in training and testing phases, without compromising overall detection performance. Furthermore, an ensem-

TABLE 8. Results of experiments performed with selected features of ADFA-LD based BoW datasets

Standard BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.6383	0.9836	0.7742	0.4811	0.7342	0.3257	0.9956
DT	0.9293	0.9344	0.9319	0.0613	0.9367	1.2667	0.0204
LR	0.9521	0.9781	0.9649	0.0425	0.9671	10.9567	0.0248
RF	0.9227	0.9781	0.9496	0.0708	0.9519	2.8898	0.0299
XGBoost	0.9626	0.9836	0.9729	0.0330	0.9747	17.2202	0.9372
AdaBoost	0.9674	0.9727	0.9700	0.0283	0.9722	15.4701	0.9412
Binary BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.6310	10.000	0.7738	0.5047	0.7291	0.9478	1.1355
DT	0.9105	0.9454	0.9276	0.0801	0.9316	1.9568	0.0190
LR	0.9278	0.9836	0.9549	0.0660	0.9569	9.0705	0.0207
RF	0.9227	0.9781	0.9496	0.0707	0.9519	4.2752	0.0437
XGBoost	0.9286	0.9945	0.9604	0.0660	0.9620	17.7116	1.1231
AdaBoost	0.9323	0.9781	0.9547	0.0613	0.9569	16.0481	1.5222
Probability BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8900	0.4863	0.6289	0.0519	0.7342	0.0181	0.6586
DT	0.9043	0.9289	0.9164	0.0849	0.9215	3.9828	0.0596
LR	0.9375	0.9836	0.9600	0.0566	0.9620	7.2691	0.0186
RF	0.9223	0.9727	0.9468	0.0708	0.9494	4.2208	0.0413
XGBoost	0.9674	0.9727	0.9700	0.0283	0.9722	16.3903	0.7978
AdaBoost	0.9672	0.9672	0.9672	0.0283	0.9696	17.4359	0.7500
TF-IDF BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.9208	0.5082	0.6549	0.0377	0.7519	0.0183	0.6826
DT	0.9144	0.9344	0.9243	0.0755	0.9291	3.0652	0.0176
LR	0.9556	0.9399	0.9477	0.0377	0.9519	7.5922	0.0139
RF	0.9188	0.9891	0.9526	0.0755	0.9544	4.1369	0.0409
XGBoost	0.9471	0.9781	0.9624	0.0472	0.9646	15.9868	0.7927
AdaBoost	0.9471	0.9781	0.9624	0.0472	0.9646	15.5812	0.9086

ble based approach with feature selection might offer better scalability and adaptability for real-time intrusion detection systems, making it a promising candidate for practical deployment. In addition, the trade-off between accuracy and processing time might be more suited to dynamic or large-scale environments where time efficiency is critical.

- It is followed that the second most successful model in terms of the ACC metric is the stacking ensemble based XGBoost model developed using the standard BoW dataset and selected features. Additionally, it appears to have better ACC values compared to the framework in [30].
- It is observed that the second most successful model in terms of the FPR metric is the framework in [30]. It is concluded that the XGBoost model, which has close FPR values with this framework and standard and probability BoW datasets and all features of these datasets are used, and the AdaBoost model, which includes all features on the standard BoW dataset, achieve better ACC values than the framework in [30]. On the other hand, the AdaBoost model using the TF-IDF BoW dataset and all its features demonstrated better performance than the framework from [30] with an FPR value of 0.0236. While most successful models in this study and the framework in [30] show similar FPR values, models in this study achieved higher accuracy, which suggests that the proposed models may offer a more efficient detection of attacks. This indicates that models in this study can deliver competitive performance with potentially more straightforward implementation compared to the framework in [30].

Table 12 compares the performance of models or HIDSs based on the ADFA-WD dataset in terms of ACC and FPR metrics. Examining Table 12, it can be observed these,

- It is concluded that the most successful model in terms of the ACC metric is XGBoost, which uses the standard BoW dataset and all its features.

TABLE 9. Results of experiments performed with all features of ADFA-WD based BoW datasets

Standard BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8384	0.9414	0.8869	0.1681	0.8845	1.9334	11.9802
DT	0.8651	0.9338	0.8982	0.1349	0.8982	5.8940	0.2458
LR	0.8276	0.9527	0.8858	0.1839	0.8818	148.8515	0.0919
RF	0.8737	0.9546	0.9124	0.1278	0.9118	34.6784	0.1223
XGBoost	0.8754	0.9565	0.9142	0.1261	0.9163	193.1389	12.0868
AdaBoost	0.8774	0.9471	0.9109	0.1226	0.9109	187.9971	9.9765
Binary BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8042	0.8696	0.8356	0.1961	0.8355	1.5192	12.8031
DT	0.8122	0.9074	0.8571	0.1944	0.8545	5.6820	0.0713
LR	0.8285	0.9225	0.8729	0.1769	0.8709	161.4618	0.0670
RF	0.8359	0.9244	0.8779	0.1681	0.8764	60.4979	0.1126
XGBoost	0.8474	0.9130	0.8789	0.1524	0.8791	276.8359	12.6199
AdaBoost	0.8411	0.9206	0.8791	0.1611	0.8782	264.7074	14.9371
Probability BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.7977	0.9093	0.8498	0.2137	0.8455	1.3783	13.4839
DT	0.8311	0.9395	0.8819	0.1769	0.8791	10.7447	0.0923
LR	0.7935	0.9301	0.8564	0.2242	0.8500	183.6092	0.0962
RF	0.8549	0.9471	0.8987	0.1489	0.8973	45.4147	0.2840
XGBoost	0.8669	0.9357	0.9000	0.1331	0.9000	222.9488	11.4506
AdaBoost	0.8641	0.9376	0.8994	0.1366	0.8991	216.5292	11.6840
TF-IDF BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8162	0.9149	0.8627	0.1909	0.8600	0.0944	12.8929
DT	0.8305	0.9357	0.8799	0.1769	0.8773	15.6639	0.1659
LR	0.7933	0.9357	0.8586	0.2259	0.8518	191.2385	0.0739
RF	0.8481	0.9395	0.8915	0.1559	0.8900	41.9776	0.1714
XGBoost	0.8613	0.9509	0.9039	0.1419	0.9027	233.4370	11.6364
AdaBoost	0.8654	0.9357	0.8992	0.1349	0.8991	234.4141	16.2237

- It can be followed that the most successful models in terms of the FPR metric are the stacking ensemble based XGBoost and AdaBoost models, developed using the standard BoW dataset and its selected features.
- It can be observed that the HIDS in [5] is less successful in terms of the FPR metric compared to the models developed in this study, as shown in Table 12.
- In terms of ACC metric, HIDS in [5] is seen to outperform the stacking ensemble based XGBoost and AdaBoost models developed with the standard BoW dataset and the selected features of this dataset by a difference of 0.0001.
- In summary, according to Table 12, it is concluded that the most successful models using ADFA-WD-based BoW datasets developed in this study generally perform better than the HIDS in [5]. With this, when comparing models in this study with the HIDS in [5], models in this study demonstrate significantly better performance in terms of FPR. Specifically, the models that utilize selected features achieve the lowest FPR values, indicating the effectiveness of the feature selection method in minimizing false positives. XGBoost and AdaBoost models developed with the standard BoW dataset and the selected features showed an ACC slightly lower by 0.0001 compared to the HIDS in [5]. This highlights the importance of the feature selection method, which leads to more accurate and reliable intrusion detection with lower FPR and similar ACC values.

The RF in [15], the framework in [30], and the HIDS in [5] were selected for comparison in this study, as they contain similar methods to those developed here. Specifically, the RF in [15] underwent preprocessing using the standard BoW method. In the preprocessing phase of the framework in [30], BoW, n-gram, and TF-IDF methods were applied. While developing the HIDS in [5], n-grams constituted part of the preprocessing phase. Moreover, another reason for selecting the RF in [15], the framework in [30], and the HIDS in [5] was that their performances had been evaluated based on the ACC and FPR metrics. Since the primary objective of this

TABLE 10. Results of experiments performed with selected features of ADFA-WD based BoW datasets

Standard BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8248	0.9433	0.8801	0.1856	0.8764	0.2395	2.5312
DT	0.8630	0.9527	0.9057	0.1401	0.9045	1.4769	0.0380
LR	0.7975	0.9527	0.8682	0.2242	0.8609	14.8021	0.0188
RF	0.8618	0.9546	0.9058	0.1419	0.9045	9.4513	0.0640
XGBoost	0.8774	0.9471	0.9109	0.1226	0.9109	30.1907	2.3641
AdaBoost	0.8774	0.9471	0.9109	0.1226	0.9109	22.7040	1.7358
Binary BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.7300	0.8998	0.8061	0.3082	0.7918	0.0097	0.2035
DT	0.7297	0.9489	0.8249	0.3257	0.8064	0.0900	0.0022
LR	0.7321	0.9659	0.8329	0.3275	0.8136	1.0438	0.0065
RF	0.7334	0.9622	0.8324	0.3239	0.8136	2.4324	0.0492
XGBoost	0.7366	0.9622	0.8344	0.3187	0.8164	1.6297	0.2364
AdaBoost	0.7362	0.9603	0.8335	0.3187	0.8155	1.6962	0.2697
Probability BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.8000	0.9074	0.8503	0.2102	0.8464	0.0082	0.1895
DT	0.7898	0.9093	0.8453	0.2242	0.8400	0.3884	0.0019
LR	0.6792	0.9527	0.7931	0.4168	0.7609	1.0609	0.0068
RF	0.7979	0.9036	0.8475	0.2119	0.8436	1.5109	0.0367
XGBoost	0.7964	0.9319	0.8589	0.2207	0.8527	4.6610	0.6152
AdaBoost	0.8049	0.8582	0.8307	0.1926	0.8318	6.9679	0.8963
TF-IDF BoW Dataset							
Model	**PRC**	**REC**	**F1**	**FPR**	**ACC**	**Training Time (s)**	**Testing Time (s)**
KNN	0.7915	0.9112	0.8471	0.2224	0.8418	0.0029	0.1675
DT	0.7763	0.9055	0.8359	0.2417	0.8291	0.3071	0.0015
LR	0.6792	0.9527	0.7931	0.4169	0.7609	0.6387	0.0046
RF	0.7823	0.9168	0.8442	0.2364	0.8373	1.7399	0.0255
XGBoost	0.7908	0.9433	0.8603	0.2312	0.8527	5.9861	0.5476
AdaBoost	0.7993	0.8809	0.8381	0.2049	0.8364	3.0592	0.2598

TABLE 11. Comparison on ADFA-LD dataset

Model/Framework	FPR	ACC
Stacking ensemble based XGBoost and AdaBoost with all features	0.0283	0.9722
Stacking ensemble based XGBoost with selected features	0.0330	0.9747
RF in [15]	0.0170	0.9840
Framework in [30]	0.0240	0.9720

TABLE 12. Comparison on ADFA-WD dataset

Model/Framework	FPR	ACC
Stacking ensemble based XGBoost with all features	0.1261	0.9163
Stacking ensemble based XGBoost and AdaBoost with selected features	0.1226	0.9109
HIDS in [5]	0.1330	0.9110

study is the development of various ML models and stacking ensemble HIDS with high ACC and low FPR values, the ACC and FPR values of similar models, frameworks, or HIDS in the literature serve as a crucial criterion for evaluating the ACC and FPR of the most successful models in this study.

On the other hand, the models developed in this study were not compared with similar models in the literature regarding training and testing time. This is because the training and testing times of the models developed in this study were measured on the same environment and hardware. In contrast, the training and testing times of similar models, frameworks, or HIDS in the literature were calculated using different environments and hardware. Therefore, it was considered neither fair nor meaningful to compare the models in this study with similar ones in the literature in terms of training and testing time. Additionally, the aim was to compare the training and testing times of the models developed using all features with those of the equivalent models that contain the selected features. Thus, the positive impact of the feature selection algorithm in Algorithm 1 on training and testing

time was observed without significantly affecting ACC and FPR values.

V. Discussion

This section provides a general review and opinion about the models developed and used methods in this study. In addition, a general discussion is made about related to HIDS.

A. General Discussion on This Study

On ADFA-based BoW datasets, stacking ensemble models developed using either all features or selected features generally achieved the highest performance in terms of ACC and FPR metrics. This is attributed to the ability of stacking ensemble models to combine the strengths of individual models. However, despite achieving the highest ACC and lowest FPR values, stacking ensemble models exhibited higher training and testing times compared to individual models due to the inclusion of multiple base models.

The feature selection method described in Algorithm 1, which combines mutual information and k-means clustering, was employed to address computational efficiency without sacrificing detection performance. This method effectively reduced the dimensionality of the datasets by eliminating redundant or less informative features while preserving those that contribute most to classification. As a result, models utilizing selected features demonstrated significantly faster training and testing times than their counterparts using all features. More importantly, the best models trained on selected features achieved comparable ACC and FPR values to those trained on the complete feature set, confirming that Algorithm 1 optimized computational efficiency and maintained detection accuracy.

In addition to feature selection, using n-grams with different BoW representations improved the system's ability to capture sequential dependencies within system/DLL call sequences, thereby enhancing intrusion detection performance and stacking ensemble models (XGBoost and AdaBoost) refined classification outcomes by leveraging multiple models, leading to higher overall accuracy and lower false positive rates. Although ensemble learning increased computational cost, its ability to integrate diverse decision boundaries from individual models resulted in superior performance across different BoW datasets.

Consequently, the proposed approach successfully balanced detection accuracy and computational efficiency. By integrating the feature selection method, n-grams, BoW methods, and ensemble learning, effective HIDSs were developed with high accuracy and low training and testing times. These findings suggest combining advanced preprocessing techniques and ensemble learning is a promising direction for enhancing HIDS in real-world applications.

B. The Most Effective Models Used in HIDS

When the performance results of individual models developed on ADFA-LD based BoW datasets are analyzed, it is observed that LR and RF models generally achieve the highest ACC values. Therefore, it is concluded that LR and RF are the most effective individual models for detecting attacks with high accuracy in HIDS designed for Linux operating systems. Moreover, RF models developed using ADFA-WD based BoW datasets have typically exhibited the highest ACC values among individual models. Thus, it is determined that RF is the most suitable individual model for achieving high accuracy in HIDS tailored for Windows operating systems. Consequently, HIDS incorporating RF models are concluded to possess higher capacities for detecting attacks with greater accuracy than those employing other individual models.

On the other hand, when all datasets are considered, stacking ensemble models have demonstrated superior performance with higher accuracy values than individual models. Among these ensemble models, XGBoost has shown higher or equivalent ACC values across all datasets compared to another ensemble model, AdaBoost. This indicates that HIDS employing XGBoost models will likely ensure operating system security with higher accuracy among stacking ensemble models.

C. Performance Comparison of Stacking Ensemble Based Models and Individual Models

Stacking ensemble based models, such as XGBoost and AdaBoost, are able to detect attacks with higher accuracy by combining the strengths of individual models. Individual models may be inadequate in learning complex attack patterns. In contrast, stacking ensemble based models are better at learning attack-specific patterns and processing features effectively. Furthermore, simultaneously generalizing both normal and attack instances may be difficult for individual models. However, normal and attack instances are learned in a balanced manner by stacking ensemble based models. As a result, better generalization ability, higher accuracy, and superior attack detection capabilities are offered by stacking ensemble based models compared to individual models.

D. Differences Between Stacking Ensemble Based XGBoost Models and Other Models

Stacking ensemble based XGBoost models across all BoW datasets are achieved the highest overall ACC and the lowest FPR values. This is attributed to the combination of the strengths of individual models and the better generalization capabilities offered by ensemble based XGBoost models. Furthermore, correcting errors from previous steps during the training phase is inherent in the structure of XGBoost models. As a result, higher accuracy is reached by XGBoost models compared to other models. On the other hand, lower performance in terms of computational cost is exhibited by stacking ensemble based XGBoost models compared to individual models. This is attributed to the fact that the training and testing phases of stacking ensemble based

XGBoost models also encompass those of the individual models.

E. Suitability of Stacking Ensemble Based Models for Real-Time Use

Stacking ensemble based XGBoost and AdaBoost models, which achieve high ACC and low FPR values, must be optimized for real-time usage. One optimization approach involves training these models with fewer features while maintaining unchanged ACC and FPR values. Another approach focuses on adjusting the models' hyperparameters to reduce computational costs. Additionally, parallel processing strategies can be implemented in these models. Consequently, these optimization methods can enhance the suitability of stacking ensemble based models for real-time scenarios.

F. The Effects of Model Accuracy on Security and Their Implemented to Large-Scale Networks

The accuracy of models used in HIDS directly influences system security. High accuracy translates explicitly to a reduction in false positives, preventing unnecessary system resource utilization due to false alarms. Additionally, high accuracy increases the likelihood of detecting genuine threats early. In other words, the potential for cyberattacks to damage the system is mitigated. Furthermore, HIDS equipped with high-accuracy models contributes to the enhanced protection of sensitive data at a higher level.

The models in HIDS that operate with high accuracy need to be adjusted to suit large-scale network environments to be used in these environments. First, the models must be scaled to cope with the ever-increasing data size and diversity. Second, the models must be able to detect attacks in real-time. Third, for the models to use the system resources efficiently, they need to be developed in a way that has low computational costs.

G. Feature Selection Method and Performance

In Algorithm 1, the feature selection method involves combining the MI method and the k-means clustering algorithm. The MI method calculates the MI values of features, and the k-means algorithm selects features with high MI values. This approach reduces the risk of overfitting by using fewer features, decreases computational cost, and improves the performance of certain models by retaining highly informative features.

When the effects of Algorithm 1 on models are examined, it is observed that for individual models developed on ADFA-LD based BoW datasets, models using all features generally achieve better ACC and FPR values compared to those using selected features. Conversely, stacking ensemble models developed with selected features perform better than those using all features in terms of ACC and FPR metrics. Overall, models utilizing all features or selected features demonstrate comparable performance in ACC and FPR metrics compared to their counterparts.

For individual and stacking ensemble models developed on ADFA-WD based BoW datasets, models using all features outperform those using selected features in terms of ACC and FPR metrics. However, models using selected features exhibit superior performance in terms of computational cost compared to their counterparts using all features. Therefore, if comparable performance is observed between models using all features and those using selected features in terms of ACC and FPR metrics, it is more appropriate to use models with selected features in HIDS for reduced computational cost. Specifically, among models developed on ADFA-LD based datasets, those using selected features are concluded to be more suitable for HIDS in Linux operating systems.

H. Use of Feature Selection Method in Different Models

The feature selection method in Algorithm 1, which combines the MI method and the k-means algorithm, can contribute to better feature selection and the development of models with higher accuracy. While integrating this feature selection method into models, the MI values of the features are first calculated, and features with high MI values are selected using the k-means algorithm. Subsequently, the chosen high-MI features are provided as input to the models. Particularly for DL models, these selected features need to undergo additional processes, such as embedding, before being fed as input. Finally, the impact of the selected features on model performance is monitored and evaluated.

I. BoW Methods and Performance

BoW methods play a crucial role in representing sequential system/DLL calls and significantly impact the performance of HIDS. Standard BoW provides a straightforward representation, making it easier for HIDS to detect frequently used attack patterns. Binary BoW enhances the ability of HIDS to capture rare system/DLL calls specific to attacks, improving the detection of unique malicious behaviors. Probability BoW creates a more sophisticated representation, allowing HIDS to better differentiate between normal and attack events. Similarly, TF-IDF BoW prioritizes system/DLL calls critical for attack detection, thereby increasing the accuracy of HIDS. These methods enable HIDS to analyze diverse patterns effectively and enhance their overall detection capabilities.

J. Effectiveness of n-gram, BoW, and TF-IDF methods on ADFA Datasets

The application of the n-gram method on ADFA datasets enables the capture of dependencies within system/DLL call sequences and the recognition of attack patterns. It also facilitates the analysis of interrelated system/DLL call transitions between sequences. As a result, n-gram contributes to HIDS achieving generally high ACC and low FPR values. On the

other hand, when only BoW methods are applied to the ADFA datasets, the frequency values of system/DLL calls within the sequences are calculated. However, BoW methods may overlook the dependencies between system/DLL call sequences. Consequently, HIDS may detect attacks at lower accuracy levels than expected. However, by combining n-gram and BoW methods, contextual relationships between system/DLL call sequences are preserved, and the frequency information of system/DLL calls is captured. This significantly enhances the performance of HIDS in detecting attacks.

On the other hand, applying only the TF-IDF method to the ADFA datasets focuses on critical system/DLL calls specific to attack-type system/DLL call sequences. However, the TF-IDF method overlooks the sequential structure within the system/DLL call sequences. Therefore, using only the TF-IDF method on the ADFA datasets may result in deficiencies in the attack detection capabilities of HIDS. However, by combining the n-gram, BoW, and TF-IDF methods, the contextual relationships within system/DLL call sequences are learned, the frequencies of system/DLL calls are calculated, and critical system/DLL calls are emphasized. As a result, HIDS that use these methods together can detect attacks with high ACC and low FPR values, as expected.

K. Limitations of ADFA Datasets

The ADFA datasets contain system/DLL call sequences from the period they were created. In other words, these datasets do not include many of the current system/DLL call sequences. This is considered a problem that may reduce the effectiveness of HIDSs developed using these datasets in the face of contemporary cyber threats. Furthermore, these datasets have an imbalance between normal and attack type system/DLL call sequences. Specifically, the number of attack type system call sequences in the ADFA-LD dataset is considerably lower than normal sequences. In contrast, the ADFA-WD dataset contains more attack-type DLL call sequences than normal ones. This imbalance in the ADFA datasets causes HIDSs developed using these datasets to be inclined to learn the majority type of system/DLL call sequences. As a result, this situation negatively affects the performance of HIDSs in detecting attacks. On the other hand, the ADFA datasets do not contain enough system/DLL call sequences, especially for training DL models. This leads to the prediction that HIDSs incorporating DL models developed with the ADFA datasets may have weak generalization abilities.

To address the issue of the outdated nature of the ADFA datasets, it is necessary to add current system/DLL call sequences to these datasets. To resolve the imbalance issue, either real system/DLL call sequences of the minority type should be added, or synthetic system/DLL call sequences should be generated and included. As a result, creating new version datasets that include current system/DLL call sequences, with reduced imbalance issues and sufficient numbers of system/DLL call sequences, remains a potential research direction.

L. Characteristics of Datasets

Various characteristics of the dataset, such as size, variability, and feature complexity, directly affect the effectiveness of feature extraction methods. The size of the dataset determines the capability of feature extraction methods. In large datasets, more accurate and reliable features can be extracted by feature extraction methods. However, feature extraction methods may identify more straightforward features with less discriminative power in smaller datasets. Therefore, the features extracted by feature extraction methods used on large datasets can enhance the models' generalization ability.

The variability within the dataset directly impacts the success of feature extraction methods. Feature extraction methods can generate a wide range of features in datasets with high variability. Therefore, selecting the most significant features among those extracted is necessary. Feature extraction methods applied to datasets with low variability tend to extract similar features, which can limit the generalization ability of the models.

The dataset's feature complexity can limit the performance of feature extraction methods. In particular, simple feature extraction methods may not be sufficiently effective on highly complex features. Therefore, it is necessary to use feature extraction methods that are suitable for the features' complexity.

In this study, 5-gram features were extracted using the n-gram method applied to the ADFA datasets. The extracted 5-gram features represent the contextual relationships within the system/DLL call sequences. The size of the raw 5-grams extracted from the ADFA datasets is quite large. For this reason, the size of the 5-grams was reduced by using the feature selection method in Algorithm 1. As a result, 5-grams that represent system/DLL call sequences and contain high information were obtained.

VI. Conclusion and Future Work

In this study, ADFA-LD and ADFA-WD datasets were first processed using n-grams in addition to the BoW methods described in [15]. As a result, standard BoW, binary BoW, probability BoW, and TF-IDF BoW datasets based on ADFA-LD and ADFA-WD with n-gram features were created. Subsequently, to reduce the dimensions of the ADFA-LD and ADFA-WD based BoW datasets and select optimal features, the MI method and k-means clustering algorithm were used together. This approach reduced the processing times of models using the dimension-reduced and selected feature BoW datasets while keeping detection and classification accuracies similar. Various individual and ensemble based ML models were then developed using all or selected features of the ADFA-LD and ADFA-WD based BoW datasets, and their performances were measured. Finally, the performances of the most successful models developed with

ADFA-LD based BoW datasets were compared with the performance of the RF model in [15] and the framework in [30]. The most successful models developed with ADFA-LD based BoW datasets performed slightly worse than the RF model in [15] regarding ACC and FPR metrics. However, these models showed better ACC performance than the framework in [30]. Also, models developed with ADFA-WD based BoW datasets and the best performance values were compared with the HIDS in [5]. The most successful models developed with ADFA-WD based BoW datasets exhibited better FPR values than the HIDS in [5]. Furthermore, the XGBoost model using the standard BoW dataset and its full features achieved better ACC values than the HIDS in [5].

A. Future Works and Research Directions Related to HIDS

Future work aims to develop hybrid HIDSs compatible with real-time environments using different feature selection methods and deep learning models. On the other hand, the future research directions that can be included in HIDS studies are as follows:

- Adjusting HIDSs to be suitable for real-time environments and evaluating their effectiveness.
- Increasing the diversity of datasets used in developing models in HIDSs and incorporating updated examples into existing datasets.
- Scaling HIDSs to reduce computational load and ensure minimal system resource consumption.
- Increasing the detection accuracy and reducing FPR values of HIDSs compared to previous HIDSs.

Acknowledgment

Zaliha Yuce Tok was supported by TUBITAK 1515 Frontier RD Laboratories Support Program.

Reviewer Appreciation

The authors would like to express their sincere gratitude to the reviewers for their meticulous evaluation and invaluable feedback, which significantly contributed to the enhancement of the quality and depth of this study.

REFERENCES

[1] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[2] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, and J. Gama, "Host-based ids: A review and open issues of an anomaly detection system in iot," *Future Generation Computer Systems*, vol. 133, pp. 95–113, 2022.

[3] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2022.

[4] H. Satilmiş, S. Akleylek, and Z. Y. Tok, "A systematic literature review on host-based intrusion detection systems," *Ieee Access*, vol. 12, pp. 27 237–27 266, 2024.

[5] Y. Kumar and B. Subba, "Stacking ensemble-based hids framework for detecting anomalous system processes in windows based operating systems using multiple word embedding," *Computers & Security*, vol. 125, p. 102961, 2023.

[6] V. Bukac, P. Tucek, and M. Deutsch, "Advances and challenges in standalone host-based intrusion detection systems," in *Trust, Privacy and Security in Digital Business: 9th International Conference, TrustBus 2012, Vienna, Austria, September 3-7, 2012. Proceedings 9*. Springer, 2012, pp. 105–117.

[7] S. Jose, D. Malathi, B. Reddy, and D. Jayaseeli, "A survey on anomaly based host intrusion detection system," in *Journal of Physics: Conference Series*, vol. 1000. IOP Publishing, 2018, p. 012049.

[8] Y. Shin and K. Kim, "Comparison of anomaly detection accuracy of host-based intrusion detection systems based on different machine learning algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, 2020.

[9] A. Shaker and S. Gore, "Importance of intrusion detection system," *Int. J. Scient. Eng. Res*, 2011.

[10] Z. Abou El Houda, B. Brik, and A. Ksentini, "Securing iiot applications in 6g and beyond using adaptive ensemble learning and zero-touch multi-resource provisioning," *Computer Communications*, vol. 216, pp. 260–273, 2024.

[11] Z. Abou El Houda, B. Brik, and L. Khoukhi, "Ensemble learning for intrusion detection in sdn-based zero touch smart grid systems," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE, 2022, pp. 149–156.

[12] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, S. J. Shepherd, and R. A. Abd-Alhameed, "An autonomous host-based intrusion detection system for android mobile devices," *Mobile Networks and Applications*, vol. 25, pp. 164–172, 2020.

[13] R. Gassais, N. Ezzati-Jivan, J. M. Fernandez, D. Aloise, and M. R. Dagenais, "Multi-level host-based intrusion detection system for internet of things," *Journal of Cloud Computing*, vol. 9, no. 1, p. 62, 2020.

[14] M. Desnoyers and M. R. Dagenais, "The lttng tracer: A low impact performance and behavior monitor for gnu/linux," in *OLS (Ottawa Linux Symposium)*, vol. 2006. Citeseer, 2006, pp. 209–224.

[15] A. A. R. Melvin, G. J. W. Kathrine, S. Pasupathi, V. Shanmuganathan, and R. Naganathan, "An ai powered system call analysis with bag of word approaches for the detection of intrusions and malware in australian defence force academy and virtual machine monitor malware attack data set," *Expert Systems*, p. e13029, 2022.

[16] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *2013 IEEE wireless communications and networking conference (WCNC)*. IEEE, 2013, pp. 4487–4492.

[17] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13 647–13 665, 2021.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.

[19] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[20] G. Creech, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," Ph.D. dissertation, UNSW Sydney, 2014.

[21] B. Subba and P. Gupta, "A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes," *Computers & Security*, vol. 100, p. 102084, 2021.

[22] J. H. Ring IV, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," *Digital Threats: Research and Practice (DTRAP)*, vol. 2, no. 4, pp. 1–29, 2021.

[23] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.

[24] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," *arXiv preprint arXiv:1611.01726*, 2016.

[25] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined cnn/rnn model," in *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*. Springer, 2019, pp. 149–158.
[26] M. Baz, "Sehids: Self evolving host-based intrusion detection system for iot networks," *Sensors*, vol. 22, no. 17, p. 6505, 2022.
[27] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
[28] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.
[29] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Canadian conference on artificial intelligence*. Springer, 2020, pp. 508–520.
[30] A. Chaudhari, B. Gohil, and U. P. Rao, "A novel hybrid framework for cloud intrusion detection system using system call sequence analysis," *Cluster Computing*, vol. 27, no. 3, pp. 3753–3769, 2024.

HAMi SATILMIŞ received the B.Sc. degree in Computer Engineering from Eskişehir Osmangazi University, Eskişehir, Turkey, in 2016 and the M.Sc. degree in Computer Engineering from Ondokuz Mayıs University, Samsun, Turkey, in 2020. Since 2020, He has continued his Ph.D. in the Department of Computational Science at Ondokuz Mayıs University. He is a research assistant with the Department of Computer Engineering at Ondokuz Mayıs University. His research interests include post-quantum cryptography, information security, machine learning, deep learning, and software engineering.

SEDAT AKLEYLEK received the B.Sc. degree in mathematics majored in computer science from Ege University, İzmir, Turkey, in 2004, and the M.Sc. and Ph.D. degrees in cryptography from Middle East Technical University, Ankara, Turkey, in 2008 and 2010, respectively. He was a Postdoctoral Researcher with the Cryptography and Computer Algebra Group, TU Darmstadt, Germany, from 2014 to 2015. He worked as a Professor at the Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Türkiye. He has been a Professor with the Department of Computer Engineering, Istinye University, Istanbul, Türkiye. He has been at the Chair of Security and Theoretical Computer Science, University of Tartu, Tartu, Estonia, since 2022. His research interests include the areas of post-quantum cryptography, algorithms and complexity, architectures for computations in finite fields, applied cryptography for cyber security, malware analysis, the IoT security, and avionics cyber security. He is a member of the Editorial Board of IEEE ACCESS, Turkish Journal of Electrical Engineering and Computer Sciences, Peerj Computer Science, and International Journal of Information Security Science.

ZALiHA YÜCE TOK received the B.S. degree in computer science, M.S. and Ph.D. degrees in cryptography from Middle East Technical University, Ankara, Turkey , in 2004, 2007 and 2016 respectively. She is currently working as an Avionic Software Engineer at ASELSAN Avionic Cyber Security Frontier Laboratory. Her current research interests include post quantum cryptography, applied cryptography for cyber security and avionic cyber security.