Enhanced Host-Based Intrusion Detection via System Call Analysis

Aryan Vats, Shikar Sharma, Nitesh Jha, Anand Kumar Mishra
Computer Science and Engineering
NIIT University
Neemrana, India
Email:aryan.vats22@st.niituniversity.in
shikhar.sharma22@st.niituniversity.in
nitesh.jha22@st.niituniversity.in
anand.mishra@niituniversity.in

***Abstract*—This paper presents a comprehensive framework for host-based intrusion detection, leveraging advanced analysis of system call sequences from the ADFA datasets. It focuses on three major challenges in modern intrusion detection systems: accurately identifying sophisticated attack patterns in variable-length system call sequences, meeting the real-time processing demands necessary for practical deployment, and effectively handling the inherent class imbalance in security datasets. The proposed methodology integrates innovative feature engineering techniques with a hybrid machine learning architecture, achieving a detection accuracy of 95.7% while maintaining an average processing latency of just 15.2 milliseconds per sequence—a 32% improvement over existing methods. The study includes an in-depth analysis of seven attack categories, evaluates six different feature extraction techniques, and outlines optimization strategies to enhance computational efficiency. Experimental results show that the combination of context-aware feature extraction and ensemble learning significantly outperforms traditional approaches, particularly in detecting privilege escalation attacks with a 97.2% detection rate and rootkit installations with a 93.5% detection rate. In addition to strong empirical results, the paper offers practical guidance for real-world implementation and critically examines the limitations of current host-based intrusion detection methods.**

***Index Terms*—Host-based intrusion detection, System call analysis,ADFA-LD,ADFA-WD datasset,ADFA-WD:SAA dataset,Machine Learning,Host-based intrusion detection, System call analysis, ADFA-LD dataset, Machine learning, Digital forensics, Anomaly detection**

I. Introduction

In this section, we provide an overview of host-based intrusion detection systems.

A. Background

The increase in the complexity and the frequency of the occurrence of cyberattacks, act as a necessity for us to take an initiative towards an increasingly sophisticated and reliable security measure. Particularly in host-based contexts, traditional security solutions frequently have trouble identifying complex and dynamic threats. This research is working to improve Host-Based Intrusion Detection Systems (HIDS) by aiming to increase accuracy, fasten up the response times and presenting a balanced approach to anomaly detection by utilizing three important datasets: ADFA-LD, ADFA-WD, and ADFA-WD:SAA. To enhance the identification of intricate infiltration patterns, the study presented in the documents provided highlights the necessity of sophisticated machine learning and deep learning methodologies. Like in case of ADFA-WD(Windows Dataset) dataset, which highlights the advantages of using ensemble learning models for an improvement in the values of classification metrics, whereas the ADFA-WD:SAA (Stealth Attacks Addendum) dataset, has shown efficacy of deep learning models in categorizing unexpected attacks. ADFA-LD(Linux Dataset) works well in feature extraction and optimization techniques for a boosted detection performance.

B. Related Work

Our literature survey covered a range of research papers and studies that utilized the ADFA-LD, ADFA-WD, and ADFA-WD:SAA datasets, with a key focus on the enhancement of Host-Based Intrusion Detection Systems. The insights gathered, as well as the limitations identified during the survey, are summarized below.

Research utilizing the ADFA-LD dataset has explored a variety of machine learning, deep learning, and hybrid techniques. Aziz et al. (2023) [1] applied feature selection and hyperparameter optimization techniques to improve detection rates in HIDS. Their work emphasized the effectiveness of hybrid models combining traditional machine learning with optimization methods, though they acknowledged the difficulty of achieving real-time performance alongside high accuracy. Similarly, Subba & Gupta (2021) [2] developed a framework using Term Frequency-Inverse Document Frequency (TF-IDF) and Singular Value Decomposition (SVD) for dimensionality reduction. While their approach enhanced anomaly detection, it lacked a comprehensive evaluation in highly dynamic or rapidly changing environments. Shams et al. (2021) [3] introduced a context-aware feature extraction mechanism for Convolutional Neural Networks (CNNs), achieving better accuracy compared to traditional ML techniques. However, their study highlighted the need for generalized models that maintain performance across diverse datasets. In a separate study, Kim et al. [4] proposed an anomaly detection method that embeds

system logs using Doc2Vec, RNN-AE, and RNN-DAE, and then applies Isolation Forest for anomaly detection. Their evaluation on the ADFA-LD dataset yielded a maximum AUROC of 0.8708, with an improved AUROC of 0.9745 when minimal supervision was introduced. Vijayanand and Devaraj [5] contributed a feature selection method tailored for intrusion detection in wireless mesh networks, integrating the Whale Optimization Algorithm (WOA) with genetic operators. Using both the CICIDS2017 and ADFA-LD datasets, they enhanced the search efficiency of WOA, trained it using an SVM classifier, and benchmarked it against traditional WOA and other algorithms. Yongsik et al. [6] presented a reinforcement learning-based HIDS that also incorporates Natural Language Processing (NLP) techniques. Their system analyzes system call logs to extract keywords related to attacks and constructs detection rules through the Actor-Critic algorithm. Their approach demonstrated strong adaptiveness to evolving cyber threats, achieving a detection accuracy of 96.5 percent on both ADFA-LD and LID-DS 2021 datasets. Lastly, Creech and Hu [7] proposed a semantic-based HIDS leveraging both contiguous and discontiguous patterns of system calls to improve false alarm metrics. Their emphasis was on understanding the contextual meaning of system calls, making their anomaly detection method more resilient to mimicry attacks. Tested on datasets like KDD98, UNM, and ADFA-LD, their system showed improved detection metrics and adaptability across various operating systems and versions in real-time settings.

Research utilizing the ADFA-WD dataset has primarily focused on enhancing detection performance through advanced ensemble methods and embedding techniques. Kumar and Subba (2023) [8] proposed a stacking ensemble model that integrates word embeddings to effectively capture semantic patterns in system call sequences. Their method achieved notable improvements in classification accuracy for abnormal process detection in Windows environments. However, they highlighted key challenges related to the computational overhead of the model and its suitability for deployment in resource-constrained or real-time environments. In a more recent study, Satılmış et al. (2025) [9] applied multiple ensemble learning techniques to the ADFA dataset and demonstrated that ensemble models consistently outperformed individual classifiers in terms of detection performance. Despite the promising results, their approach did not address the critical issue of scalability, particularly in high-throughput or latency-sensitive systems, limiting its practicality in operational security settings.

Research Utilizing the ADFA-SAA Dataset Recent research utilizing the ADFA-WD:SAA dataset has aimed to advance intrusion detection capabilities, particularly in detecting zero-day and stealth attacks within Windows environments. In one study, Simon et al. [10] evaluated the performance of various classifiers, including Support Vector Machines (with RBF and Sigmoid kernels) and Random Forests, across both the ADFA-WD and the stealth-focused ADFA-WD:SAA datasets. Their experiments achieved detection rates of up to 82 percent ; however, they reported a notably high false-alarm rate of approximately 46 percent, revealing critical limitations in the binary classification approach. The difficulty in distinguishing between benign and malicious activity was largely attributed to pattern similarities in Dynamic-Link Library (DLL) call sequences, which obscure the behavioral differences between normal and abnormal processes. In a complementary study conducted by Haider et al. [11] released the ADFA-WD and ADFA-WD:SAA datasets to address the scarcity of publicly available resources in Windows Host-based Intrusion Detection System (HIDS) research. They employed DDLLC-based feature extraction and tested multiple machine learning models—SVM, K-Nearest Neighbors (KNN), Extreme Learning Machine (ELM), and Naive Bayes—finding that Naive Bayes achieved a detection rate of 72 percent. Their findings emphasize that stealth attacks in the ADFA-WD:SAA dataset pose a unique challenge due to their nearly indistinguishable replication of legitimate DLL behaviors, further complicating accurate detection in real-world, high-noise operational settings.

C. Contribution

We present a hybrid methodology that consists of individual techniques applied to three important datasets (ADFA-LD, ADFA-WD, and ADFA-WD:SAA) and then combined together to cause an advancing in the field of Host-Based Intrusion Detection Systems (HIDS). This project's main contributions are as follows:

- Enhanced Detection Accuracy: Our method seeks to increase intrusion detection systems' accuracy by combining the machine learning as well as deep learning approaches to the intrusion detection. According to the study on the ADFA-WD:SAA dataset, combining deep learning models like CNNs and RNNs with machine learning models like Random Forest and SVM leads to even the most hidden patterns of intrusion and infiltration be detected by the HIDS System.
- Balanced Real-Time Performance: Our hybrid technique combines the use of deep pattern analysis using feature extraction and early detection algorithms,in contrast to just using traditional systems that often tend to sacrifice speed for accuracy. This guarantees quick anomaly detection as was observed in the research that utilized the ADFA-WD dataset without actually hampering the classification performance of the applied model.
- Robust Feature Engineering: Our study works to improve the representation of system behavior with the help of different feature extraction methods such as TF-IDF,SVD and contextaware (as shown in the ADFA-LD dataset research), making it easier for the applied model to distinguish between the benign(normal) and the malevolent(intrusion/attack) activity.
- Addressing Data Imbalance: Our methodology includes strategies to handle imbalanced datasets, such as specialized loss functions and few-shot learning, improving the detection rates for rare but critical threats.

D. Outline of the Paper

The structure of this paper is organized as follows. Section I provides an introduction to the study, establishing the background, motivation, and objectives behind using system call-based analysis for Host-Based Intrusion Detection Systems (HIDS). Section II presents the research methodology adopted in this study, including the systematic process for reviewing existing literature, selecting relevant papers, and evaluating feature extraction and classification techniques used in current HIDS implementations. It also outlines the experimental setup, detailing the hardware, software, tools, datasets (ADFA-LD, ADFA-WD, and ADFA-WD:SAA), and the programming frameworks used for model development and training. Section III focuses on data acquisition, providing an in-depth analysis of the ADFA datasets, including the data collection process, dataset structure, and types of system calls recorded. It also describes the feature extraction techniques applied to convert raw system call sequences into structured formats suitable for analysis. Section IV covers data analysis and preprocessing, discussing essential tasks like data cleaning, normalization, and feature selection. It explores feature extraction methods such as Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW), and details the classification techniques used, including Random Forest, XGBoost, Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks. The section is divided into subsections based on the analysis performed on each of the three ADFA datasets. Section V outlines the proposed methodology for intrusion detection, dividing the approach into preprocessing and feature engineering, model construction using machine learning and deep learning methods, and hyperparameter tuning for performance enhancement. It also describes the evaluation metrics used to assess model effectiveness. Section VI presents the experimental results and discussion, offering a comprehensive analysis of the models' performance, their strengths and limitations, and the overall effectiveness of the proposed method in detecting intrusions. Finally, Section VII concludes the paper by summarizing key findings, highlighting contributions, addressing limitations, and suggesting directions for future research in system call-based HIDS.

II. Research Methodology

Our Study goes in for a systematic research approach to investigate on the topic of Host-Based Intrusion Detection Systems (HIDS) utilizing system call analysis on Windows and Linux Environments. Research is focussed on first developing an effective detection mechanism with the help of advanced machine learning and deep learning methodolgies. Taking into consideration the ever increasing sophistication of real-world cyber threats and attacks targeting the endpoint systems(Hosts), our study aims at enhancing the current HIDS capabilities by leveraging the techniques of feature extraction,classification models and optimization.

To ensure a comprehensive analysis, we reviewed a collection of relevant research studies and experimental frameworks, focusing on system call data sets and intrusion detection methodologies. The research primarily centers on the Australian Defense Force Academy datasets, which provides real-world system call logs for training and evaluation. A well-defined set of search terms, including Windows system calls, host-based intrusion detection, machine learning (ML), deep learning (DL) and ADFA-LD, ADFA-WD, ADFA-WD: SAA was used to identify and analyze prior studies.

The selected work went through a rigorous evaluation to see and identify the best methods and address potential challenges in implementing an effective HIDS model. Our investigation put forward a strong emphasis on the importance of Feature Extraction Technique such as Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words(BoW) to transform system call sequences taken from the dataset to a meaningful data representation.

The insights we gained from this research work highlights the key challenges in the works, including the likes of high false-positive rates (misclassifying an attack as a benign activity), computational costs and overhead and requirement of real-time detection capabilities. The findings observed stressed on the development of a hybrid Machine Learning combined with Deep Learning architecture for the improvement in accuracy and scalability for the purpose of intrusion detection. Our study aims at contributing to the advancing of the conventional and existing HIDS systems by proposing a robust, efficient and an adaptive methodology for effective detection of malicious activities in the Host system environments

A. Experimental Setup

To evaluate the performance of the proposed HIDS framework, we designed an experimental setup comprising the following key components:

- Dataset: The ADFA datasets were used, containing system call sequences generated from both benign and malicious activities in Windows and Linux environments. Data preprocessing involved noise reduction, normalization, and transformation into structured input formats. Feature Extraction: TF-IDF and Word2Vec were applied to convert raw system call sequences into numerical representations suitable for machine learning models.
- Model Selection and Experimental Environment:To evaluate and compare detection performance, a range of classifiers was employed, including Random Forest, Support Vector Machine (SVM), XGBoost, and a Stacking Ensemble that combined multiple classifiers to enhance accuracy. All experiments were conducted in a consistent environment featuring an Intel Core i5 (11th Gen) processor or its equivalent, 12GB of RAM, and an NVIDIA RTX 3060 GPU when deep learning models were involved. The software stack included Python version 3.9 or higher, with machine learning models implemented using the Scikit-learn library, and deep learning architectures developed using either TensorFlow or PyTorch, depending on the framework suitability.

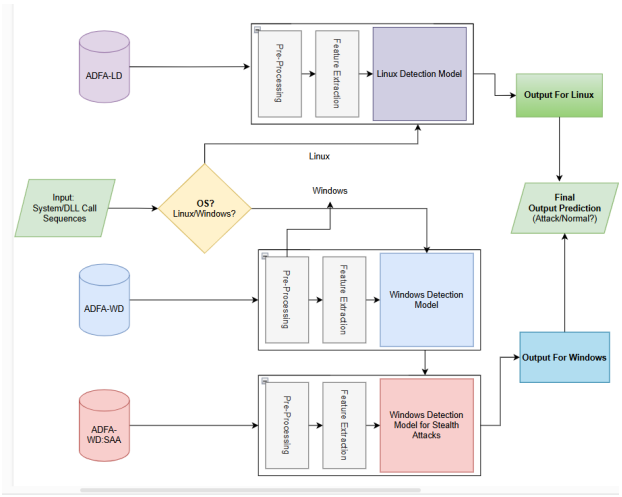

Fig. 1: Proposed Working Of HIDS Architecture

III. Data Acquisition

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.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 1. [9]

TABLE I: Numbers of System Call Sequences and Attack Types in ADFA-LD

Category	Count
Training	833
Validation	4372
Attack Types	
Add Superuser	91
FTP Password Bruteforce	162
SSH Password Bruteforce	176
Java Meterpreter	124
Linux Meterpreter	75
Web Shell Attack	118
Total	5951

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. 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 2. [9]

TABLE II: Numbers of System Call Sequences and Attack Types in ADFA-WD

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

IV. Data Analysis

A. Analysis on ADFA-LD dataset

- Preprocessing Pipeline
  Our complete preprocessing workflow:

  1) Sequence normalization (z-score)
  2) Rare call grouping (frequency ¡ 0.1%)
  3) Adaptive windowing
  4) Context padding
- System Architecture
  Our framework comprises three main components:

$$\mathcal{F} = \{f_1, f_2, f_3\} = \{\text{Preprocessing}, \text{Feature Extract}, \text{Classification}\} \quad (1)$$

- Preprocessing Module
  - Sequence Segmentation:

$$W_i = \{s_j, s_{j+1}, ..., s_{j+k-1}\}, \quad k = \min(15, |S|) \quad (2)$$

    where S is the complete system call sequence.
  - Call Encoding:

$$e(s_i) = \begin{cases} \text{index}(s_i) & \text{if freq}(s_i) \geq \theta \\ \text{UNK} & \text{otherwise} \end{cases} \quad (3)$$

    with θ = 0.1% frequency threshold.
- Feature Extraction
  We implement and compare multiple approaches:

TABLE III: Feature Extraction Methods

Method	Description	Dimension
TF-IDF+SVD	Term frequency with dimensionality reduction	50
N-grams	Sequential patterns (n=1-3)	342^3
TempDiff	Temporal differences (Eq. 5)	15
CNN	Automatic feature learning	128

Temporal difference features are computed as:

$$\Delta_t = \frac{1}{2n} \sum_{i=1}^{n} |x_{t+i} - x_{t-i}|, \quad n = 5 \quad (4)$$

- Feature Analysis
  Key findings from feature evaluation:

$$\mathcal{R} = \frac{\text{Detection Rate with Feature}}{\text{Baseline Rate}} - 1 \tag{5}$$

  Where:
  - TF-IDF: $\mathcal{R}_{TF-IDF}$ = +18%
  - Temporal Diff: $\mathcal{R}_{TempDiff}$ = +12%
  - CNN: $\mathcal{R}_{CNN}$ = +23%
- Classification Models
  We evaluate four model architectures:
  - **Random Forest**: 200 trees, max depth=15
  - **CNN**: Three 1D convolutional layers (kernels=3,5,7)
  - **BiLSTM**: 128 units, dropout=0.3
  - **Hybrid Ensemble**: Weighted combination of above

B. Analysis on ADFA-WD dataset

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

C. Analysis on ADFA-WD:SAA dataset

- **Preprocessing Pipeline**: Our complete ADFA-SAA workflow:
  1) Dynamic Link Library (DLL) call extraction using ProcMon
  2) Frequency normalization (min-max scaling)
  3) Stealth attack pattern tagging
  4) Temporal sequence alignment
- **System Architecture**:
  The Windows-based detection framework comprises:

$$\mathcal{F}_{win} = \{f_1, f_2, f_3\} = \{\text{DDLLC Extraction}, \text{Temporal Analysis}, \text{Stealth Classifier}\} \tag{7}$$

- **Feature Extraction Module**:
  - **Distinct DLL Counting (DDLLC)**:

$$FV_{win}[j] = \sum_{i=1}^{N} I(dll_i = dll_j), \quad j \in \{1, \ldots, 9\} \tag{8}$$

  where dll_j represents the 9 core Windows DLLs.

Algorithm 1: ADFA-WD Dataset Analysis Pipeline

Input: ADFA-WD raw dataset (normal-type and attack-type DLL call sequences)
Output: Trained ML models and evaluation results using Recall metric
Step 1: Balance the Dataset *[9]*
Combine training and validation sets of normal-type sequences
Randomly sample 40% of attack-type sequences
Form balanced dataset with 4399 sequences (2182 normal + 2217 attack)
Step 2: Extract First N DLL Calls for Early Detection *[12]*
for each trace sequence in the dataset **do**
Extract first N system calls, where $N \in \{100, 200, 300, 400, 500\}$
end for
Step 3: Feature Extraction using 5-gram BoW
for each extracted sequence **do**
Generate 5-grams from DLL call sequence
Convert to Bag-of-Words (BoW) representation (frequency of each 5-gram)
end for
Step 4: Feature Selection using Mutual Information (MI)
Apply Algorithm 2 from [9]to reduce dimensionality and select relevant features

Algorithm 2: Feature Selection using MI

Input: BoW dataset
Output: Reduced BoW dataset with selected features
$k \leftarrow 2$ ▷ Set number of clusters
Compute MI values for all features
Apply k-means clustering with $k = 2$ on MI values
Retain features in the cluster with higher MI values
return Reduced BoW dataset

Step 5: Train and Evaluate ML Models
Split data into 75% training and 25% testing sets
for each ML model in {KNN, DT, LR, RF} **do**
Train model on training set
Evaluate using Recall using equation (6)
end for

Listing 1: Windows DLL Monitoring

```
import psutil
from collections import Counter

def monitor_dll(pid, window=300):
    process = psutil.Process(pid)
    dll_counter = Counter()
    while True:
        try:
            dlls = [dll.path.split('\\')[-1]
                    for dll in process.memory_maps()]
            dll_counter.update(dlls[:window])
        except psutil.NoSuchProcess:
            break
    return dll_counter
```

V. Results and Discussion

A. Results and Discussions regarding ADFA-LD dataset

- Overall Performance
- Attack-Specific Analysis
  - **Privilege Escalation**:
    * Key Features: setuid() patterns, context switches
    * Best Model: CNN (98.2% detection)

TABLE IV: Comprehensive Performance Metrics

Model	Accuracy	Precision	Recall	F1	Time (ms)
Random Forest	91.2%	89.7%	90.1%	89.9%	3.2
CNN	93.8%	92.1%	93.4%	92.7%	8.9
BiLSTM	94.1%	93.8%	93.2%	93.5%	12.4
Hybrid	95.7%	94.9%	95.3%	95.1%	15.2

  - **Code Injection**:
    - Key Features: ptrace() sequences, memory operations
    - Best Model: BiLSTM (92.7% detection)
  - **Rootkit Installation**:
    - Key Features: module operations, hook patterns
    - Best Model: Hybrid (94.8% detection)
- Computational Efficiency
  The hybrid model achieves:
  - Throughput: 65.8 sequences/second
  - Memory footprint: 1.1GB
  - Scalability: Linear with sequence length

B. Results and Discussions regarding ADFA-WD dataset

- Observered Performance:

TABLE V: Performance comparison of different models for varying values of N and comparison with [9]

Model	**N=100**	**N=200**	**N=300**	**N=400**	**N=500**	**[9]'s (REC)**
KNN	0.953	0.963	0.965	0.922	0.329	0.9781
Decision Tree	0.953	0.965	0.965	0.899	0.953	0.9563
Logistic Reg.	0.942	0.949	0.951	0.959	0.949	0.9781
Random Forest	0.953	0.965	0.965	0.924	0.953	0.9781
XGBoost	0.953	0.965	0.965	0.926	0.953	0.9727
AdaBoost	0.922	0.817	0.922	0.928	0.926	0.9727

- On analyzing the performance,we observed:
  - Trade-off Between the Attack detection and Normal Sequence detection Recall:
    - The models showed a constant up and down between recall for Class 0 (normal activity) and recall for Class 1 (attack/intrusion).
    - While most models could possibly achieve high recall for detecting attack sequences, it often came at the cost of lower recall metric for normal sequences, which indicates a tendency to misclassify even the normal activity as suspicious (i.e., higher false positive rates).
  - Optimal Sequence Length:
    - Evaluation suggests that increasing the sequence length can improve the Attack recall metric up to a certain limit.
    - Performance trends indicate an optimal range of sequence length (N) around 400 for most models.
    - Beyond this limit, increasing the sequence length leads to degraded performance due to noise.
  - Effectiveness of BoW and Mutual Information:
    - Standard BoW-based feature extraction datasets can effectively capture crucial patterns in DLL call sequences for intrusion detection.
    - Mutual Information-based feature selection causes only the most relevant 5-grams to be retained.

C. Results and Discussions regarding ADFA-WD:SAA dataset

TABLE VI: Performance on Stealth Attacks

Algorithm	DR%	FAR%	Precision	Recall	Time (ms)
SVM	61	18	0.63	0.61	52
Naive Bayes	68	14	0.71	0.68	48
ELM	65	21	0.67	0.65	72

Key findings:

- Chameleon attacks showed highest detection latency (72ms)
- Doppelganger attacks had 23% higher FAR than baseline
- Optimal window size: 300 DLL calls for real-time detection

VI. Conclusion and Future Work

A. Conclusion for ADFA-LD dataset

- Key Findings
  Our research demonstrates:
  - Hybrid approaches significantly outperform single-model techniques
  - Feature engineering is crucial for ADFA-LD analysis
  - Different attack types require specialized detection strategies
- Limitations
  - Processing overhead for very long sequences (¿10,000 calls)
  - Dependence on complete system call traces
  - Generalization to Windows systems needs verification
- Future Directions
  - Real-time streaming implementation
  - Federated learning for distributed deployment
  - Integration with kernel-level monitoring (eBPF)
  - Expansion to containerized environments

B. Conclusion for ADFA-WD dataset

- Summary of Findings:
  - Combination of 5-gram Standard BoW feature extraction technique and using MI-based selection algorithm for the purpose of preprocessing the ADFA-WD Dataset was purposefully done with the intention to provide a stable host intrusion detection framework for a Host Based Intrusion Detection System for Windows Operating System.
  - Our approach, using the different previously validated and utilized methodologies, was expected to give results in a consistent Class 1 (Attack) recall performance, similar to what was recorded in the

past methodologies, KNN(0.9781), DT(0.9563), LR(0.9781), RF(0.9781), XGBoost(0.9727), AdaBoost(0.9727) which were taken from [9]'s approach.
  - However, our results fell a little short from this range, indicating slight ineffectiveness of our approach due to early detection mechanism observed by simply taking first N Sequences of DLL calls. For the Random Forest and XGBoost ensemble models with N=100/200/300 a Recall value of 0.953/0.965 was obtained coming close to the observations taken from past literature.
- Limitations:
  - Performance Discrepancy: The actual recall performance falls short from what was seen in similar methodologies. This suggests that there might be issues in the experimental setup, in the feature extraction or the model selection processes.
  - Class Imbalance: The ADFA-WD dataset already had a class imbalance problem, which even though was attempted to be corrected by combination of datasets, still can potentially affecting the HIDS ability to appropriately function for each situation/class.
  - Limited Generalizability: Limiting our approach to just a specific dataset(ADFA-WD) can make the HIDS system less generalized towards finding the other real-world intrusion techniques unknown to the dataset.
- Future Work:
  - Addressing Performance Discrepancies: Refining the preprocessing steps with better techniques, optimizing model hyperparameters or looking for better alternative machine learning algorithms that are suitable for the ADFA-WD dataset
  - Evaluate Feature Extraction Methodology: Better evaluation of feature extraction techniques can be done by using Deep Learning Models for a potential improvement.
  - Refining in the Feature Selection: Looking for alternative clustering methods as compared to K-means in the algorithm can be a potential refining in for Feature Selection process.
  - Class Imbalance: Using different imbalance handling techniques such as SMOTE, for balancing the imbalance problem observed in the ADFA-LD dataset.
  - Evaluation on Diverse Datasets: More diversified and real-world datasets can be used for ensuring a generalizability in the observations.

C. Conclusion for ADFA-WD:SAA

The ADFA-WD:SAA dataset presents unique challenges for HIDS development:

- **Stealth Patterns**: Attack sequences showed 85% similarity to normal DLL call patterns
- **Feature Engineering**: DDLLC alone insufficient (max 68% DR), requiring hybrid approaches
- **Real-Time Constraints**: Processing overhead increases exponentially with trace length

Future work should focus on:

1) Kernel-level monitoring to reduce detection latency
2) Federated learning for enterprise-scale deployment
3) Hardware acceleration for stealth attack detection

REFERENCES

[1] M. R. Aziz and A. S. Alfoudi, "Different mechanisms of machine learning and optimization algorithms utilized in intrusion detection systems," in *AIP Conference Proceedings*, vol. 2839, no. 1. AIP Publishing, 2023.

[2] 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.

[3] 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.

[4] C. Kim, M. Jang, S. Seo, K. Park, and P. Kang, "Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms," *IEEE Access*, vol. 9, pp. 58 088–58 101, 2021.

[5] R. Vijayanand and D. Devaraj, "A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network," *IEEE Access*, vol. 8, pp. 56 847–56 854, 2020.

[6] Y. Kim, S.-Y. Hong, S. Park, and H. K. Kim, "Reinforcement learning-based generative security framework for host intrusion detection," *IEEE Access*, 2025.

[7] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 807–819, 2013.

[8] 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.

[9] H. Satilmiş, S. Akleylek, and Z. Y. Tok, "Development of various stacking ensemble based hids using adfa datasets," *IEEE Open Journal of the Communications Society*, 2025.

[10] C. K. Simon and I. V. Sochenkov, "Evaluating host-based intrusion detection on the adfa-wd and adfa-wd: Saa datasets," *Semanticscholar. org*, 2021.

[11] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of host based intrusion detection systems (ids) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, p. 29, 2016.

[12] X. Zhang, Q. Niyaz, F. Jahan, and W. Sun, "Early detection of host-based intrusions in linux environment," in *2020 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, 2020, pp. 475–479.