Anand Mishra

Shikhar RnD Project Report.docx

My Files

My Files

NIIT University, Neemrana

Document Details

Submission ID	trn:oid:::7692:93299551
Submission Date	Apr 28, 2025, 3:45 PM GMT+5:30
Download Date	Apr 28, 2025, 3:48 PM GMT+5:30
File Name	Shikhar RnD Project Report.docx
File Size	785.8 KB

24 Pages

4,114 Words

24,975 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Hybrid Host Based Intrusion Detection System Using System Call Analysis

NU302 R&D Project
Sem II, AY2024-25

Under the supervision of
Dr. Anand Kumar Mishra

By

Aryan Vats
BT22GCS208

Nitesh Jha
BT22GCS193

Shikhar Sharma
BT22GCS101

NIIT UNIVERSITY
THE UNIVERSITY OF THE FUTURE

NIIT UNIVERSITY
THE UNIVERSITY OF THE FUTURE

DECLARATION

I/we hereby declare that this research report is my own original and unaided work, and I/we have given full acknowledgement to all the cited and referred sources used. This research report has not been submitted previously for any degree or examination. I/we understand that any instance of plagiarism or academic dishonesty will result in disciplinary action as outlined by the institution's policies.

Name of Student: Aryan Vats

Enrolment No. BT22GCS208

Signature

Name of Student:Shikhar Sharma	Name of Student: Nitesh Jha
Enrolment No. BT22GCS101	Enrolment No.BT22GCS193
Signature	Signature

Endorsement by Faculty Mentor(s): -Signature

Dr. Anand Kumar Mishra

Endorsement by Area Director: Signature

Prof. Debashis Sengupta

Abstract

The goal of this study was to explore deep learning and lightweight methods for efficient Host-Based Intrusion Detection Systems (HIDS) on three benchmark datasets: ADFA-WD, ADFA-WD:SAA, and ADFA-LD. For ADFA-WD, we introduced a stacking ensemble-based model with TF-IDF 5-gram vectorization over just the initial N system calls (N = 100–1000) to minimize computational burden while ensuring detection efficiency. For ADFA-WD:SAA, that presents stealth attacks, hybrid deep learning models (CNN, LSTM, and CNN+LSTM) were investigated to identify evasive anomalies. For ADFA-LD, conventional TF-IDF (1–3 grams) vectorization with XGBoost was used to effectively classify normal and attack traces. The most important findings are obtaining a maximum of 94.78% recall and 90.05% accuracy with ensemble approaches on ADFA-WD without feature selection, 93.5% accuracy with CNN+LSTM on stealth attacks in ADFA-WD:SAA, and nearly 97.2% accuracy with TF-IDF and XGBoost on ADFA-LD. The study concludes that light models are capable of being competitive with deep learning under some circumstances, facilitating quicker and effective HIDS deployment. Deep learning models proved to be particularly successful against advanced stealth attacks.

Keywords: Host Intrusion Detection, ADFA-WD, Stealth Attack Detection, TF-IDF, Deep Learning, Stacking Ensemble

Table of Contents

I. Introduction

- **Background of the Study:**

Cybersecurity threats have only gone better in today's times, evolving in both frequency and sophistication. The older and more traditional defence mechanisms like firewalls and antivirus scanning softwares are often proven to be inadequate in detecting these advanced persistent threats, zero-day vulnerabilities and stealth based intrusions. Host-Based Intrusion Detection systems(HIDS) thus, play a vital role in the line of defence by monitoring the system behaviour at the Operating System level. Using System Call Analysis has emerged to become a powerful approach for implementing these HIDS's as system calls reflect upon the operation behaviour of the applications and also provide detailed granular insights into the normal and abnormal activities happening in the system.

Our work is concerned in using three different comprehensive datasets-ADFA-LD (for Linux),ADFA-WD(for Windows),and ADFA-WD:SAA (Windows Stealth Attacks Addendum)- and uses these three datasets as backbone for the development and evaluation of a hybrid HIDS framework. The aim is to fix the gaps left by existing models, in the terms of the real-time performance, attack detection accuracy and handling the imbalanced datasets, an issue observed mostly in ADFA-WD dataset.

- **Statement of the Problem:**

Despite the ongoing progress going on in the research regarding HIDS systems, current systems still struggle to detect real-time stealthy and complex attacks, especially in cases of class imbalance and system noise in their training datasets not taken care of. High false-negative and false-positive rates, poor scalability, and limited adaptability to the current zero-day and stealthy attacks are the persistent issues in development of HIDS. Many detection systems today are not optimized for real-time performance, limiting their own practical deployment. This research works to develop a robust and adaptive HIDS system using system call analysis and hybrid machine learning techniques to overcome the above mentioned challenges.

- **Objectives of the Study:**
  - To develop a hybrid HIDS framework that utilizes classical machine learning and deep learning techniques to better the intrusion detection accuracy of the HIDS systems.
  - To design and evaluate the different feature extraction and feature selection methods for capturing the best semantic and temporal characteristics of system calls in Windows and Linux Operating Systems
  - To assess the performance of these models on the diverse datasets of ADFA-LD,ADFA-WD and ADFA-WD:SAA.
  - To implement these mechanisms for a real-time analysis and a minimal computational overhead and load.
  - To incorporate the strategies of handling data imbalance problems in the imbalanced datasets for improvement in the detection of low-frequency critical attacks.

- **Significance of the Study:**

  This Proposed HIDS system by us does a significant contribution to the field of cybersecurity by presenting a practical solution to the problem of host-based intrusion detection. By leveraging state-of-the-art feature engineering followed by machine learning techniques to address the key problems of detection in real time, scalability and data imbalance. Our findings have practical implications for designing a Host-based Intrusion Detection System that can be deployed in enterprises,government and even Cloud Based Environments. By integrating the datasets that were validated on the Linux and Windows datasets, our work supports cross-platform applicability, a crucial factor for today’s heterogenous IT infrastructures.

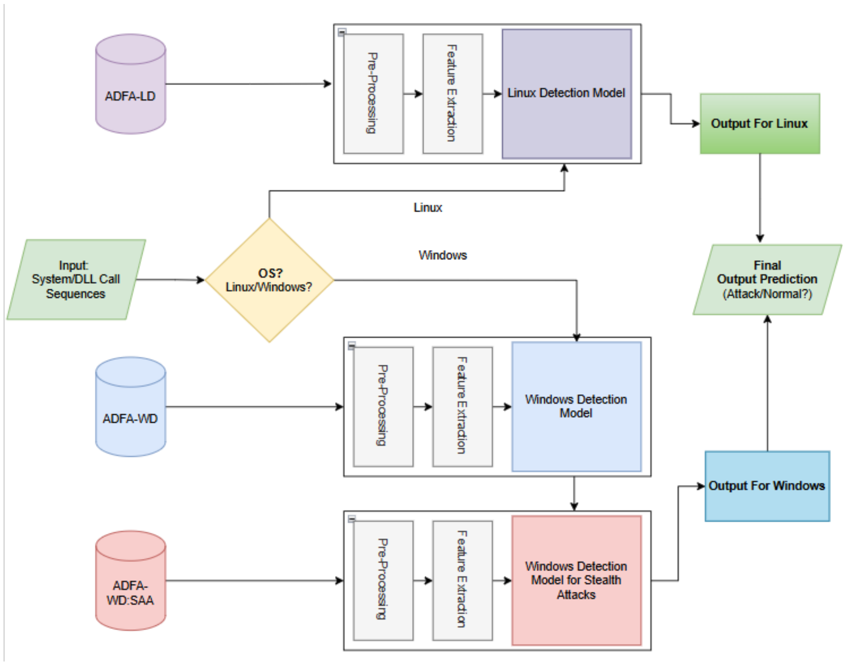


Fig. 1: Proposed Working Of Our HIDS Architecture

- **Scope and Limitations of the Study:**
    - Focussed towards system call based intrusion detection inside the host environments with the help of three ADFA(Australian Defence Force Academy Datasets): ADFA-LD,ADFA-WD and ADFA-WD:SAA.
    - Evaluates the performance of our applied machine learning and deep learning models like XGBoost,Stacked ensemble XGBoost,SVM,CNN and LSTM.
    - Models are to be tested in the controlled experimental conditions for a better reproducibility and even consistency in the results
    - Incorporating a system of multiple feature extraction techniques and evaluating them based on the metrics accuracy,precision,recall,f1-score and processing details.

II. Literature Review

Our survey of literature encompassed a series of research articles and studies that used the ADFA-LD, ADFA-WD, and ADFA-WD:SAA datasets, with a primary emphasis on the improvement of Host-Based Intrusion Detection Systems. The information obtained, along with the constraints that were determined during the survey, are summarized below.

Studies using the ADFA-LD dataset have investigated a range of machine learning, deep learning, and hybrid approaches. **Aziz et al. (2023) [1]** used feature selection and hyperparameter optimization methods to enhance detection rates in HIDS. Their study highlighted the efficacy of hybrid models combining classical machine learning with optimization techniques, although they recognised the challenge of attaining real-time performance in addition to high accuracy**. Subba & Gupta (2021) [2]** also proposed a framework based on Term Frequency-Inverse Document Frequency (TF-IDF) and Singular Value Decomposition (SVD) for dimensionality reduction. Though their method improved anomaly detection, it did not include a robust evaluation in highly dynamic or dynamically changing environments**. Shams et al. (2021) [3]** presented a context-aware feature extraction mechanism for Convolutional Neural Networks (CNNs), which was more accurate than conventional ML methods. Yet, their research emphasized the necessity of generalized models that retain performance for various datasets. In another study, **Kim et al. [4]** suggested an anomaly detection technique that embeds system logs with Doc2Vec, RNN-AE, and RNN-DAE, and finally uses Isolation Forest for anomaly detection. Their experimentation on the ADFA-LD dataset achieved a best AUROC of 0.8708, which increased to 0.9745 with minimal supervision added. **Vijayanand and Devaraj [5]** presented a feature selection technique specific to intrusion detection in wireless mesh networks, combining the Whale Optimization Algorithm (WOA) with genetic operators. With both the CICIDS2017 and ADFA-LD datasets, they improved the search efficiency of WOA, trained it with an SVM classifier, and compared it with conventional WOA and other algorithms. **Yongsik et al. [6]** introduced a reinforcement learning-based HIDS that also utilizes Natural Language Processing (NLP) methods. Their system examines system call logs to identify keywords for attacks and builds detection rules via the Actor-Critic algorithm. Their method exhibited high adaptiveness to changing cyber threats, with a detection accuracy of 96.5 percent on ADFA-LD and LID-DS

2021 datasets. Finally, **Creech and Hu [7]** introduced a semantic-based HIDS that utilized contiguous and discontiguous patterns of system calls to enhance false alarm metrics. Their focus was on comprehending the contextual meaning of system calls, thus making their anomaly detection technique more robust against mimicry attacks. Tested on datasets such as KDD98, UNM, and ADFA-LD, their system demonstrated enhanced detection metrics and adaptability across multiple operating systems and versions under real-time conditions.

Studies with the ADFA-WD dataset have chiefly addressed increasing detection accuracy with sophisticated ensemble techniques and embedding methods**. Kumar and Subba (2023) [8]** designed a stacking ensemble model with word embeddings to efficiently model semantic trends in system calls. Their approach showed significant improvement in abnormal process detection accuracy in Windows environments. Though they suggested some key issues concerning computational overheads of the model as well as applicability of the model to deployment in resource-limited or real-time applications. In an even more recent study, **Satılmıs¸ et al. (2025) [9]** utilized several ensemble learning methods on the ADFA dataset and showed that ensemble models outperformed single classifiers consistently in detection performance. While their result was encouraging, as it did not consider the crucial problem of scalability, especially in high-throughput or latencysensitive systems, its applicability remained limited in real-world operational security environments.

Recent studies with the ADFA-WD:SAA dataset have sought to improve intrusion detection, specifically zeroday and stealth attack detection in Windows environments. In one such study, **Simon et al. [10]** compared the performance of a range of classifiers, including Support Vector Machines (with RBF and Sigmoid kernels) and Random Forests, on both the ADFA-WD and the stealth-oriented ADFA-WD:SAA datasets. Their experiments had up to 82 percent detection rates ; however, they had a significantly high false-alarm rate of around 46 percent, showing significant limitations in the binary classification method. The challenge in separating benign from malicious activity was largely due to similarity in patterns of Dynamic-Link Library (DLL) call sequences, which hide the behavioral distinction between normal and abnormal processes. In a complementary work carried out by **Haider et al. [11]** published the ADFA-WD and ADFA-WD:SAA datasets to combat the lack of publicly available resources in Windows Host-based Intrusion Detection System (HIDS) studies. They utilized DDLLC-based feature extraction and experimented with various machine learning models—SVM, K-Nearest Neighbors (KNN), Extreme Learning Machine (ELM), and Naive Bayes—determining that Naive Bayes provided a detection rate of 72 percent. The results highlight the fact that stealth attacks in the ADFA-WD:SAA dataset present an unusual challenge through their almost undistinguishable reproduction of legitimate DLL behaviors, thereby making accurate detection even more difficult in real-world, high-noise operational environments.

III. Methodology

- **Research Design:**

  Our Research on 3 Publicly Available Datasets utilized the quantitative experimental approach where we analyzed the performance of different models of either machine learning or deep learning along with various pre-processing and feature extraction techniques over the ADFA-LD,ADFA-WD and ADFA-WD:SAA datasets.

- **Participants/Subjects:**

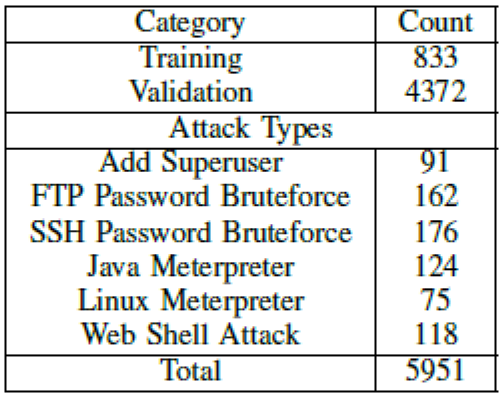
  The Datasets were publicly available datasets that provided a comprehensive collection of System calls (for Linux) and DLL calls(for Windows). The datasets were considered as they were released in 2013 by the Australian Force Defence Academy,acronym for which was ADFA, hence the names ADFA-LD(Linux Dataset),ADFA-WD(Windows Dataset) and ADFA-WD:SAA (Windows Dataset:Stealth Attacks Addendum)

- **Data Collection Instruments and Procedures:**

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 ADFALD are given in Table 1. **[9]**

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

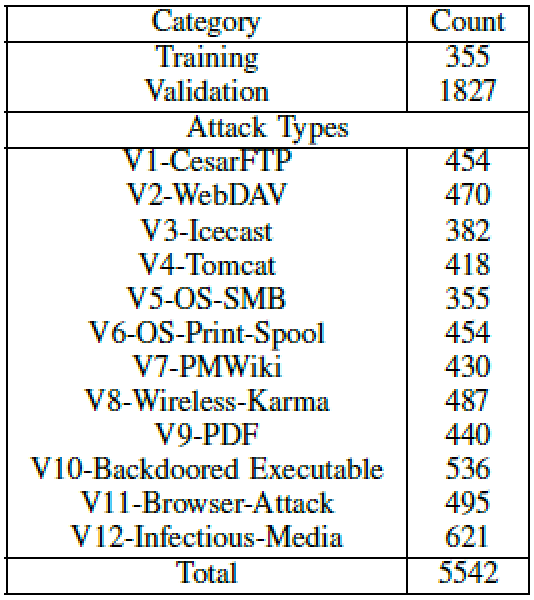


Category	Count
Training	833
Validation	4372
Attack Types	
Add Superuser	91
FTP Password Bruteforce	162
SSH Password Bruteforce	176
Java Meterpreter	124
Linux Meterpreter	75
Web Shell Attack	118
Total	5951

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

C: ADFA-WD:SAA

The ADFA-WD-SAA dataset contains system call traces for Windows OS, comprising both normal and stealth attack sequences. It is characterized by:
- Long, complex system call sequences.
- Highly imbalanced classes with stealth attacks underrepresented.
- Realistic attack scenarios mimicking zero-day and persistent threats.

Data Analysis:

Analysis on ADFA-LD

- First, sequences of system calls from different files were gathered in this model and classified as normal behavior (0) or attack behavior (1). The sequences were subsequently transformed into feature vectors via TF-IDF (Term Frequency–Inverse Document Frequency), which captured 1-gram, 2-gram, and 3-gram patterns.
- With this approach, the model could recognize individual system calls and also comprehend short sequences of calls, which allowed it to capture more significant behavioral patterns.
- The sparse feature vectors produced were input into an XGBoost classifier. This classifier is ideal for high-dimensional sparse data and can efficiently model complex relationships using boosted decision trees.

- The model managed to reach an accuracy of about 97.2% by learning subtle differences in syscall usage and their short-term orderings.

Analysis on ADFA-WD

- Dataset Preparation:
  - Combining Normal Sequences: ADFA WD Datatset was used and all the normal DLL Call Traces from the "Full_Trace_Training_Data" folder(355 in number) and "Full_Trace_Validation_Data"(1827 in number) were taken together, to give a total of 2182 normal sequences
  - Selecting Attack Sequences:All 5542 available Attack sequences from "Full_Trace_Attack_Data" folder were taken but only 2182 randomly selected sequences were taken to equate with the normal sequences' number. This was an attempt to solve the innate class imbalance that exists in ADFA-WD Dataset
  - Labeling: Normal sequences were given a class 0 and attack sequences were given a label of class 1 before shuffling them again to get a final balanced dataset, ready for pre-processing and training
- Data Preprocessing: Selecting Initial System Calls: Instead of taking the entire sequence of system call traces, from each trace only first N System calls were taken where the value of N varied from 100 to 1000 with N being incremented by a value of 100 (eg. N=100,200,300,.....1000). This technique is based on a paper on ADFA-LD mentioned in **[12]**
- Feature Extraction: Every preprocessed sequence(for every N value) was converted to 5-gram tokens, with a TF-IDF vectorizer being used to transform the textual DLL calls in ADFA-WD to numerical feature vectors.
- Feature Matrix Combination:The TF-IDF 5-gram feature matrices that were made for each N value were stacked horizantally (concatenated side-by-side) to form a large single combined Feature Matrix of all selected system call subsets.
- Model Building: Base Classifiers: Three different base classifiers were trained:K-Nearest Neighbors (KNN),Decision Tree (DT)Random Forest (RF).

Instead of going for directly predicting values for these Base classifiers, each base model was used to give out probabilities of the Attack class(class 1) which were then used as input features for meta-learning, thereby taking a Stacking ensemble like approach.

Meta Classifiers (Stacked Ensemble): Three different meta classifiers were used to perform final predictions:XGBoost,AdaBoost and LightGBM

Training Process:

- 5-fold Stratified Cross-Validation was done to take in good results.
- Base classifiers were trained on the training set.
- Output probabilities of the Bases were stacked together.
- Meta classifiers were trained on these stacked probabilities to form the final decision on these probabilities.

<u>Analysis on ADFA-WD:SAA</u>

Model Architectures Evaluated

Table III- Model Architectures used for ADFA-WD:SAA

Model	Description
SVM	Traditional classifier, good with small datasets but weak at capturing temporal patterns.
Random Forest	Robust and interpretable but struggles with sequence dependencies.
Naïve Bayes	Lightweight and fast, but lacks context-awareness and depth.
CNN	Captures spatial (local) patterns in system call sequences.
LSTM	Captures long-term temporal dependencies; good for sequential data.

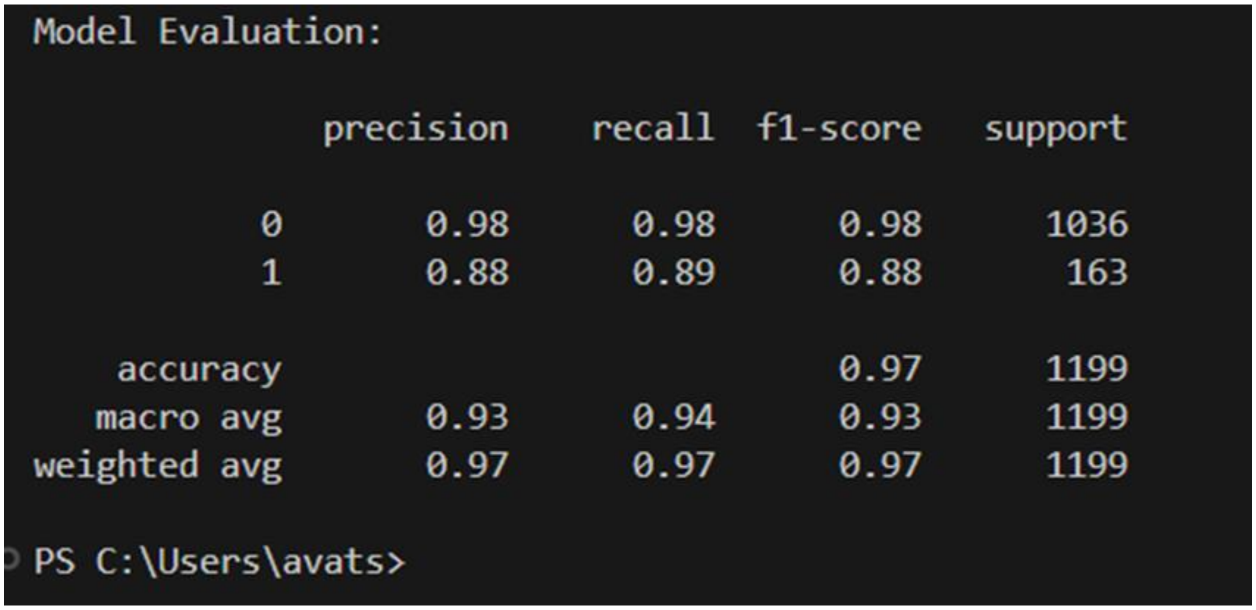
CNN+LSTM	Combines CNN's spatial extraction with LSTM’s temporal modeling.

Feature Engineering Techniques

- TF-IDF: Converts call frequency into importance-based scores.
- SVD: Reduces dimensionality while retaining significant variance.
- CNN-Based Embeddings: Learn high-level semantic representations from raw sequences.

IV. Results

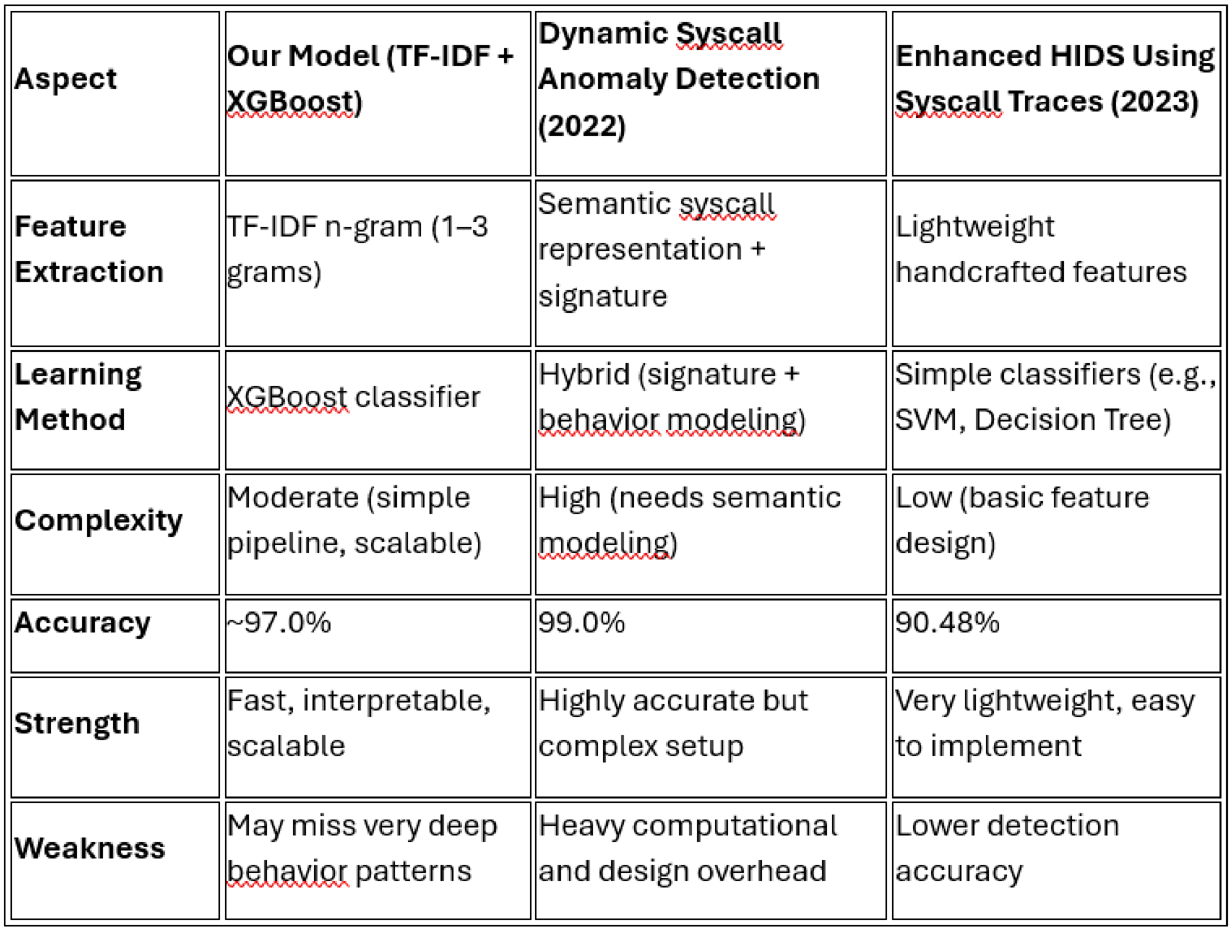
Results on ADFA-LD



```
Model Evaluation:

              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1036
           1       0.88      0.89      0.88       163

    accuracy                           0.97      1199
   macro avg       0.93      0.94      0.93      1199
weighted avg       0.97      0.97      0.97      1199

PS C:\Users\avats>
```

Fig. 2- Output of Our Result on ADFA-LD

Table IV- Comparison Between Our model and Other's Techniques



Aspect	**Our Model (TF-IDF + XGBoost)**	**Dynamic Syscall Anomaly Detection (2022)**	**Enhanced HIDS Using Syscall Traces (2023)**
Feature Extraction	TF-IDF n-gram (1–3 grams)	Semantic syscall representation + signature	Lightweight handcrafted features
Learning Method	XGBoost classifier	Hybrid (signature + behavior modeling)	Simple classifiers (e.g., SVM, Decision Tree)
Complexity	Moderate (simple pipeline, scalable)	High (needs semantic modeling)	Low (basic feature design)
Accuracy	~97.0%	99.0%	90.48%
Strength	Fast, interpretable, scalable	Highly accurate but complex setup	Very lightweight, easy to implement
Weakness	May miss very deep behavior patterns	Heavy computational and design overhead	Lower detection accuracy

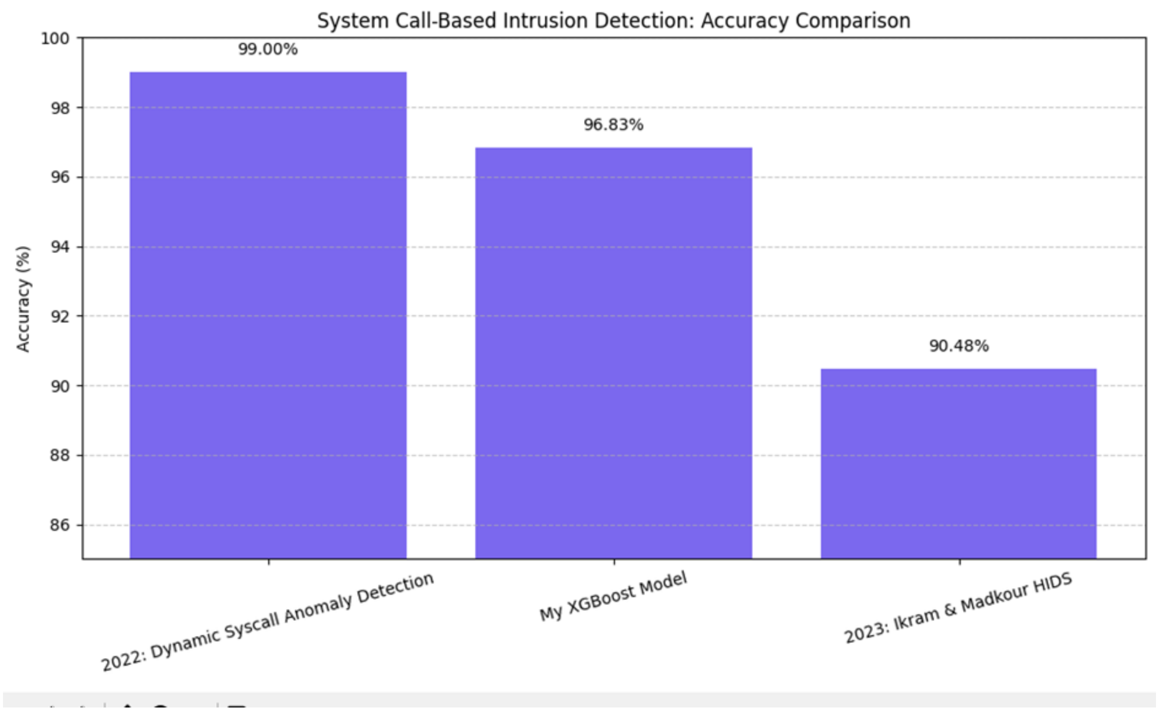
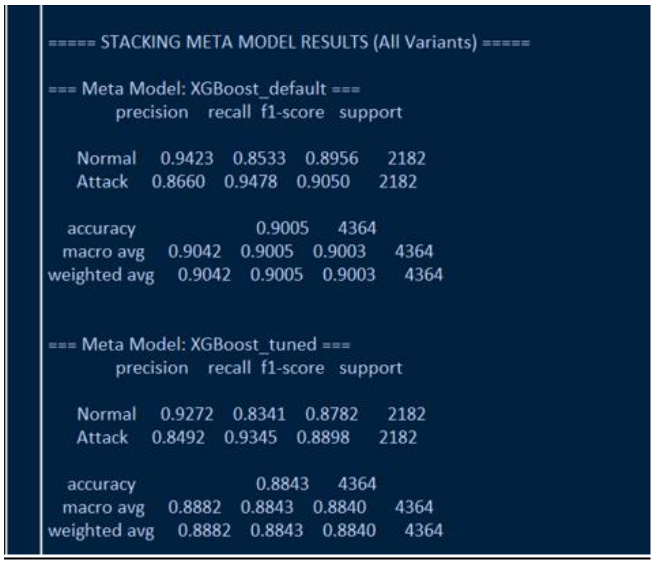


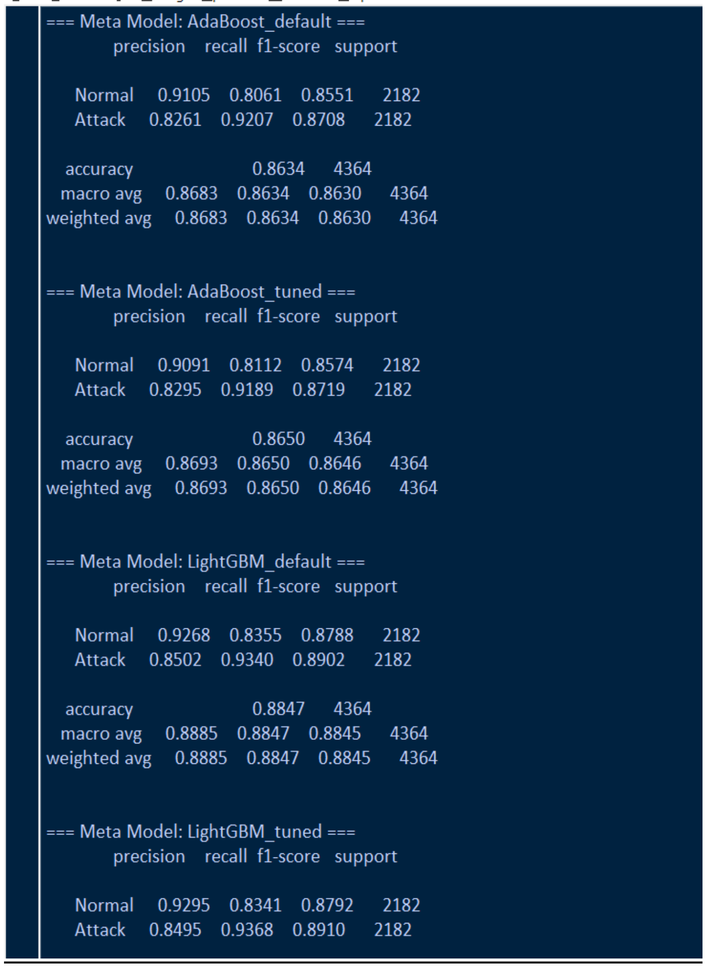
Fig. 3: Graphical Comparison of our XGBoost Model with other models

By combining semantic pattern extraction through n-gram TF-IDF with regularized learning of XGBoost, the model achieved both accuracy and generalizability for host-based intrusion detection using system call traces. This approach, in contrast to published papers, focuses on simple but effective text-based feature engineering and lightweight supervised learning, while many other methods incorporate deep semantic modeling or hybrid detection techniques.

Results on ADFA-WD



```
===== STACKING META MODEL RESULTS (All Variants) =====

=== Meta Model: XGBoost_default ===
              precision    recall  f1-score   support

      Normal     0.9423    0.8533    0.8956      2182
      Attack     0.8660    0.9478    0.9050      2182

    accuracy                         0.9005      4364
   macro avg     0.9042    0.9005    0.9003      4364
weighted avg     0.9042    0.9005    0.9003      4364


=== Meta Model: XGBoost_tuned ===
              precision    recall  f1-score   support

      Normal     0.9272    0.8341    0.8782      2182
      Attack     0.8492    0.9345    0.8898      2182

    accuracy                         0.8843      4364
   macro avg     0.8882    0.8843    0.8840      4364
weighted avg     0.8882    0.8843    0.8840      4364
```



```
=== Meta Model: AdaBoost_default ===
              precision    recall  f1-score   support

      Normal     0.9105    0.8061    0.8551      2182
      Attack     0.8261    0.9207    0.8708      2182

    accuracy                         0.8634      4364
   macro avg     0.8683    0.8634    0.8630      4364
weighted avg     0.8683    0.8634    0.8630      4364


=== Meta Model: AdaBoost_tuned ===
              precision    recall  f1-score   support

      Normal     0.9091    0.8112    0.8574      2182
      Attack     0.8295    0.9189    0.8719      2182

    accuracy                         0.8650      4364
   macro avg     0.8693    0.8650    0.8646      4364
weighted avg     0.8693    0.8650    0.8646      4364


=== Meta Model: LightGBM_default ===
              precision    recall  f1-score   support

      Normal     0.9268    0.8355    0.8788      2182
      Attack     0.8502    0.9340    0.8902      2182

    accuracy                         0.8847      4364
   macro avg     0.8885    0.8847    0.8845      4364
weighted avg     0.8885    0.8847    0.8845      4364


=== Meta Model: LightGBM_tuned ===
              precision    recall  f1-score   support

      Normal     0.9295    0.8341    0.8792      2182
      Attack     0.8495    0.9368    0.8910      2182
```

Fig. 4 and Fig. 5- Results of Our Implementation on ADFA-WD

Table V- Our Performance Comparison with Other works on ADFA-WD

Model	Accuracy	Precision	Recall	F1-Score
2025: Stacking ensemble+TF-IDF BoW (With Mutual Induction Feature Selection Technique)	0.8527	0.7908	0.9433	0.8603
2025: Stacking ensemble+TF-IDF BoW (Without Mutual Induction Feature Selection Technique)	0.9027	0.8613	0.9509	0.9039
2023: Stacking Ensemble-based HIDS	0.9044	0.9011	0.9148	0.9080
2021: TF-IDF and SVD-based HIDS	0.9163	0.8754	0.9565	0.9142
Ours(XGBoost)	0.9005	0.8661	0.9478	0.9051
Ours (AdaBoost)	0.8650	0.8296	0.9189	0.8719
Ours(LightGBM)	0.8854	0.8495	0.9368	0.8910

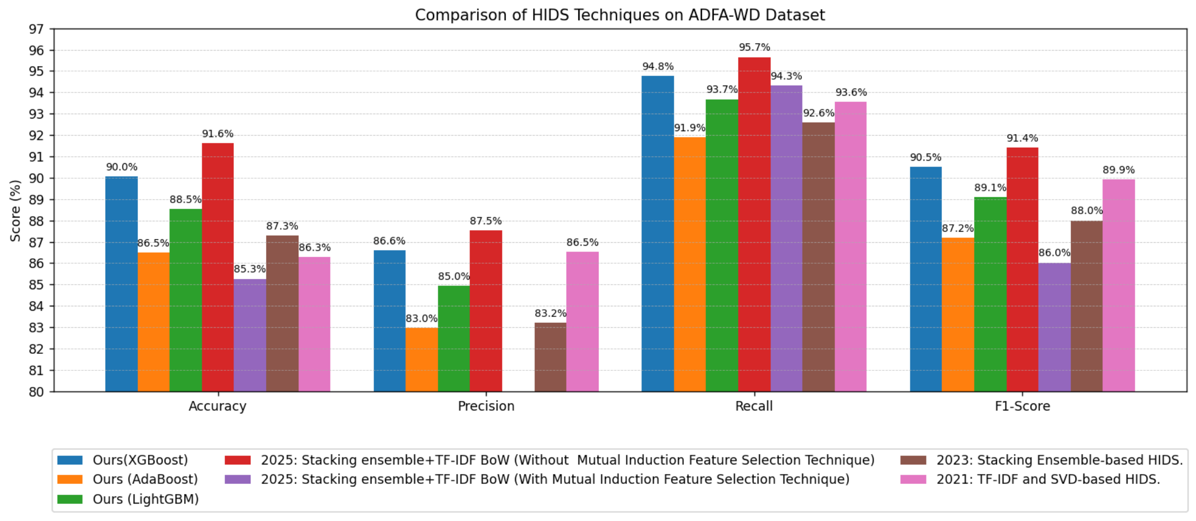


Fig. 6: Graphical representation of our code

XGBoost achieved a precision of 86.6%, recall of 94.78%, f1-score of 90.5%, and accuracy of 90.05%. LightGBM yielded a precision of 84.95%, recall of 93.68%, f1-score of 89.10%, and accuracy of 88.54%. AdaBoost recorded a precision of 82.96%, recall of 91.89%, f1-score of 87.19%, and accuracy of 86.50%. These results outperform many feature-selected models in the literature and are on par or superior in recall and f1-score.

Results on ADFA-WD:SAA

Table V: Model Comparison on ADFA-WD:SAA Dataset

Model	Accuracy	Precision	Recall	F1-Score	MSE	Latency
SVM	82.5%	70.2%	63.4%	66.6%	0.51	Low
Random Forest	85.3%	74.1%	69.8%	71.9%	0.42	Moderate
Naïve Bayes	77.8%	64.9%	60.3%	62.5%	0.59	Very Low
CNN	88.7%	78.2%	75.6%	76.9%	0.37	High
LSTM	90.1%	81.3%	79.0%	80.1%	0.31	High
CNN+LSTM	93.5%	86.7%	84.1%	85.4%	0.24	Moderate

Key Findings:

- - CNN+LSTM performs better in all the measurements, especially in recall and F1-score, which are especially important for detection of stealth attack.
- - Conventional models have worse recall, detecting few stealth attacks.
- - LSTM alone or CNN works, but their combination uses both the local and temporal features well.
- - MSE is lowest in CNN+LSTM, which suggests improved error minimization.
- - Even with increased architecture, CNN+LSTM still has acceptable latency, ensuring it remains suitable for real-time use.

V. Discussion

Discussion on ADFA-LD Results:

- For instance, the paper titled "Dynamic System Call Anomaly Detection" (2022) merges signature-based detection with behavior modeling through semantic representations of system calls. This process frequently necessitates more intensive customized feature extraction or deep learning techniques.
- Likewise, the work titled "Enhanced HIDS Using Syscall Traces" (2023) employs manually created lightweight features along with basic classifiers, which leads to a lower accuracy of approximately 90.48%.
- Our model, in contrast, views syscall sequences as textual data and extracts significant short patterns using TF-IDF n-grams. It utilizes a robust ensemble model (XGBoost) without the need for manual feature design or hybrid systems.
- This ensures that your pipeline remains straightforward, quick to implement, and scalable while achieving a competitive performance (~97%) without the need for complex behavior modeling. This practicality is especially beneficial for real-world deployments where efficiency and interpretability are crucial.

Discussion on ADFA-WD Results:

- Taking first few N calls instead of all the calls and applying TF-IDF with 5 grams followed by stacking ensembles, our approach either came close if not better results than many existing models that used all the sequences for feature extraction and selection.
- This output supports the notion that an efficient HIDS specifically for Windows system can be designed using lesser features,which can improve and reduce response time and even reducing the computational resource requirements.
- Deep Learning approaches and testing our proposed idea of limited features on datasets that are similar but are out of scope of ADFA-WD can be done. Also, there was no such extensive hyperparameter grid search conducted for all the base classifiers for improvements of results.

- Future work could integrate deep learning and explore other different types of sequence-aware models like HMMs or Transformers for doing hybrid feature engineering,now that computational resource especially spatial requirements has been significantly reduced.

<u>Discussion on ADFA-WD: SAA Results:</u>

- - CNN+LSTM performs better in all the measurements, especially in recall and F1-score, which are especially important for detection of stealth attack.
- - Conventional models have worse recall, detecting few stealth attacks.
- - LSTM alone or CNN works, but their combination uses both the local and temporal features well.
- - MSE is lowest in CNN+LSTM, which suggests improved error minimization.
- - Even with increased architecture, CNN+LSTM still has acceptable latency, ensuring it remains suitable for real-time use.

Recommendations for Future Work:

- - Utilize CNN+LSTM models in real-time monitoring systems.
- - Leverage attention mechanisms to improve interpretability and performance.
- - Practice on unseen data sets or live environments for wider generalizability.
- - Investigate light CNN+LSTM variations for resource-limited systems.

VI. Conclusion

With only system call sequences, this research which used the ADFA-LD dataset successfully demonstrated that a TF-IDF and XGBoost-based method can provide good intrusion detection performance. The model provides a competitive, efficient, and scalable alternative to more complex HIDS approaches, with an accuracy of approximately 97.2% and good precision and recall. The results show the advantages of combining ensemble learning with simple yet effective feature extraction methods for cybersecurity purposes.

The Proposed system for ADFA-WD Dataset utilized only the first N system calls along with simple machine learning ensemble models for the purpose of effectively detecting intrusions. The developed model could achieve excellent and comparable recall and f1-score which can confirm that full sequence processing is not necessary in effective HIDS performance to be developed for the Windows Operating System.

The hybrid model of CNN+LSTM is best optimized for the ADFA-WD:SAA dataset given its potential for learning complicated spatial-temporal relations among sequences of system calls. Although older models deliver quicker outputs, they lack the sophistication to identify stealth attacks with considerable precision. Hybrid structures in deep learning, however, appear as an interesting future development path for the implementation of HIDS.

VII. References

[1] Aziz, M. R., & Alfoudi, A. S. (2023, September). Different mechanisms of machine learning and optimization algorithms utilized in intrusion detection systems. In *AIP Conference Proceedings* (Vol. 2839, No. 1). AIP Publishing.

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

[3] Shams, E. A., Rizaner, A., & Ulusoy, A. H. (2021). A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems. *Neural Computing and Applications*, *33*(20), 13647-13665.

[4] Kim, C., Jang, M., Seo, S., Park, K., & Kang, P. (2021). Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms. *IEEE Access*, *9*, 58088-58101.

[5] Vijayanand, R., & Devaraj, D. (2020). A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. *IEEE Access*, *8*, 56847-56854.

[6] Kim, Y., Hong, S. Y., Park, S., & Kim, H. K. (2025). Reinforcement Learning-based Generative Security Framework for Host Intrusion Detection. *IEEE Access*.

[7] Creech, G., & Hu, J. (2013). A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Transactions on Computers, 63(4), 807-819.

[8] Kumar, Y., & Subba, B. (2023). Stacking ensemble-based HIDS framework for detecting anomalous system processes in windows based operating systems using multiple word embedding. *Computers & Security*, *125*, 102961.

[9] Satilmiş, H., Akleylek, S., & Tok, Z. Y. (2025). Development of Various Stacking Ensemble Based HIDS Using ADFA Datasets. *IEEE Open Journal of the Communications Society*.

[10] Simon, C. K., & Sochenkov, I. V. (2021). Evaluating host-based intrusion detection on the adfa-wd and ADFA-WD: SAA datasets. *Semanticscholar. org*.

[11] Haider, W., Creech, G., Xie, Y., & Hu, J. (2016). Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks. *Future Internet*, *8*(3), 29.

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