Indian Institute of Technology, Ropar

PGSL Project Report: LSTM Based IDS on ADFA-LD Dataset

Instructor: Dr. Basant Subba

Name: Amber Gupta Entry No: 2024AIM1001

Name: Yash Narnaware Entry No: 2024AIM1011

Name: Prince Raj Entry No: 2024AIM1012

Name: Aksshay Mathew P Entry No: 2024AIM1013

Contents

Chapter 1

Introduction

The increasing prevalence and sophistication of cyber-attacks pose significant challenges to system security, necessitating advanced methods for detecting and responding to threats in real-time. Traditional signature-based intrusion detection systems (IDS) struggle to recognize novel or evolving attack patterns, highlighting the need for more robust, adaptable, and intelligent detection methods. In response, deep learning models—especially those leveraging sequential data—have gained considerable attention due to their ability to model complex patterns and identify subtle anomalies in data sequences.

This project focuses on the design and implementation of a Long Short-Term Memory (LSTM)-based classifier to detect given system call sequence is normal or some intrusion, a core component of Host Intrusion Detection Systems (HIDS). By analyzing system calls generated by applications and operating systems, our model aims to identify attack behaviors early and accurately, enhancing the detection capabilities of IDS.

The LSTM model architecture was chosen for its ability to capture temporal dependencies in sequential data, making it well-suited for system call analysis. Additionally, this project explores the integration of n-grams and Word2Vec embeddings to preprocess system call sequences, transforming them into structured, machine-readable formats while preserving contextual information. For experimentation, the ADFA Linux Dataset (ADFA-LD) was utilized, containing various types of attacks and corresponding system call sequences. This dataset provided a foundation for training and testing our model, with attack types including brute force, meterpreter-based, and shell-based attacks.

Chapter 2

Dataset

The dataset we have used to build an intrusion detection system(IDS) is ADFA-LD [1]. It is a dataset containing system call sequences while doing everyday tasks on a Linux-based system, and we also have system call sequences when someone is trying to attack the system. ADFA-LD dataset is used for anomaly detection; our objective is to classify whether a given system call sequence is normal.

The dataset contains three folders: Attack Data Master, Training Data Master, and Validation Data Master. Attack Data Master contains system call sequences when a particular attack occurs. Training Data Master and Validation Data Master contain the regular system call sequences captured during everyday tasks on a Linux system.

Data Type	Trace Count
Normal Training Data	833 Traces
Normal Validation Data	4373 Traces
Attack Data	746 Traces

Table 2.1: Trace count in dataset

The Attack Data folder has six subfolders for different attack types. We have less attack data to simulate the real-world scenario where we would have more normal system calls than attack system calls.

In the validation dataset, we only have traces of normal system calls, not attack system calls, because the motivation behind using the ADFA-LD dataset is to detect any anomaly in system calls rather than identify the specific attack type. We can have many types of attacks on our system, so rather than learning the pattern of each attack, we focus on learning the pattern of normal system calls, and if a given system call does not fit in the normal system call pattern, we label it as an attack(intrusion detected).

2.1 Introduction to Word2Vec

Word2Vec is a popular technique in Natural Language Processing (NLP) that represents words as dense vectors, capturing semantic and syntactic relationships between words. Developed by Google, Word2Vec leverages two main architectures: Continuous Bag of Words (CBOW) and Skip-gram.

2.1.1 CBOW and Skip-gram Architectures

- **Continuous Bag of Words (CBOW):** The CBOW model predicts the current word based on the context words (surrounding words). It takes the average of context word vectors and tries to predict the target word. This approach is efficient and works well with a large corpus.
- **Skip-gram:** The Skip-gram model, on the other hand, predicts the context words given the current word. It is particularly effective in capturing rare word associations and learning robust representations for infrequent words.

2.2 Applying Word2Vec in Host-based Intrusion Detection Systems (HIDS)

Word2Vec embeddings can be effectively applied to encode sequences in logs or network events, transforming them into numerical vectors that represent the semantic meaning of system calls or log messages.

2.2.1 Enhanced Context Understanding

By capturing the short-term dependencies between system calls or log events, Word2Vec embeddings can identify patterns in normal and anomalous behavior. Specifically, using n-grams as input provides better local context, which enhances the model's understanding of short-term dependencies in the data.

2.2.2 Improving Detection with LSTM and Word2Vec

When combined with Long Short-Term Memory (LSTM) networks, Word2Vec embeddings enhance the model's ability to learn meaningful sequences in intrusion data. The LSTM captures long-term dependencies, while Word2Vec provides a dense, context-aware representation of each event or word. This combination improves the ability to detect subtle patterns indicative of anomalies or intrusions.

$$\text{Embedding} = \text{Word2Vec}(SystemCall\ Sequence)$$

$$\text{Output} = \text{LSTM}(\text{Embedding})$$

This integration helps in creating a powerful Host-based Intrusion Detection System (HIDS) that learns from the sequential patterns of system calls or logs and detects unusual behaviors more effectively.

Chapter 3

LSTM based binary classifier

The main objective of using the ADFA-LD dataset is to classify whether a given system call sequence is normal. For that, we build a bi-directional LSTM-based binary classifier.

3.1 Data Preparation

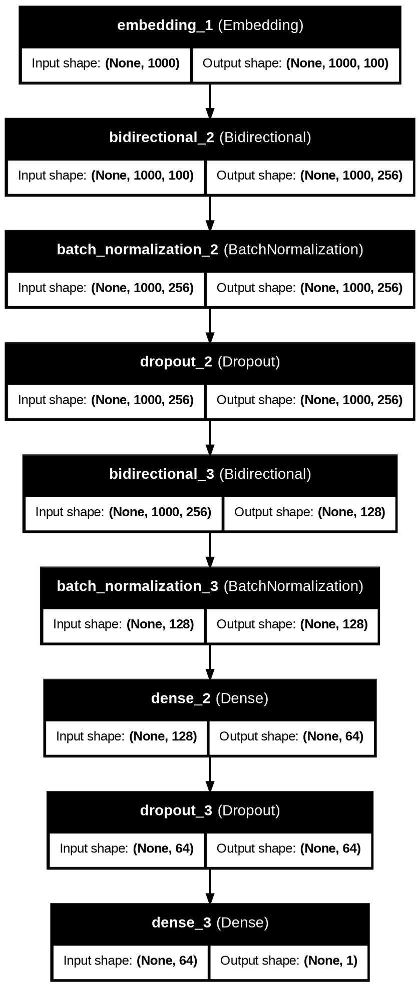
In the binary classifier's case, we mark all nonnormal calls' labels as 1. So, our training dataset will look like this -

Data type	**Label in dataset**	**Number of traces**
Normal	0	833
Attack(all types included)	1	746

Table 3.1: Attack-wise trace count and labels in dataset for binary classification

Instead of passing system call sequences directly, we trained the Word2Vec model on n-grams of system calls to generate embeddings of system calls.

3.2 Model architecture



embedding_1 (Embedding)
Input shape: (None, 1000)
Output shape: (None, 1000, 100)
bidirectional_2 (Bidirectional)
Input shape: (None, 1000, 100)
Output shape: (None, 1000, 256)
batch_normalization_2 (BatchNormalization)
Input shape: (None, 1000, 256)
Output shape: (None, 1000, 256)
dropout_2 (Dropout)
Input shape: (None, 1000, 256)
Output shape: (None, 1000, 256)
bidirectional_3 (Bidirectional)
Input shape: (None, 1000, 256)
Output shape: (None, 128)
batch_normalization_3 (BatchNormalization)
Input shape: (None, 128)
Output shape: (None, 128)
dense_2 (Dense)
Input shape: (None, 128)
Output shape: (None, 64)
dropout_3 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
dense_3 (Dense)
Input shape: (None, 64)
Output shape: (None, 1)

Figure 1: model architecture

We used various techniques like batch normalization and dropout layers to improve the accuracy of the model. Using bi-directional LSTM instead of normal LSTM improved the model accuracy significantly. On a validation dataset, we are getting accuracy in the range of $81 - 83\%$. We have tried a simplified version of the current model. We tried a more complex architecture model(with more neurons and dense layers) but did not find it helpful. Overfitting occurs in a more complex model, resulting in poor validation loss. In this architecture, if we train it for more than 15 epochs, we see some signs of overfitting; that is why we are training it for 15 epochs.

Chapter 4

LSTM based multi-class classifier

4.1 Dataset Preparation

The ADFA-LD training dataset has 833 normal traces and 746 attack traces, . Amongst 746 traces, the distribution of the number of traces for each attack and the label that we assigned them is as follows -

Attack Name	**Label in dataset**	**Number of traces**
Normal(not attack)	0	833
Web Shell	1	118
Meterpreter	2	75
Hydra SSH	3	176
Hydra FTP	4	162
Adduser	5	91
Java Meterpreter	6	124

Table 4.1: Attack-wise trace count and labels in dataset

The number of traces for each class is not evenly distributed, which can cause problems during the model training, resulting in low performance. This phenomenon is called class imbalance.

To solve this problem, we will do the oversampling of minority classes and the undersampling of the majority class. We cannot just oversample minority classes to the number of traces of the majority class; in that case, the model will overfit.

We oversampled the minority classes(duplicated the rows) so that each minority class has 280 number of traces and undersampled the majority class(normal traces) to have 350 number of traces now.

4.2 Model architecture

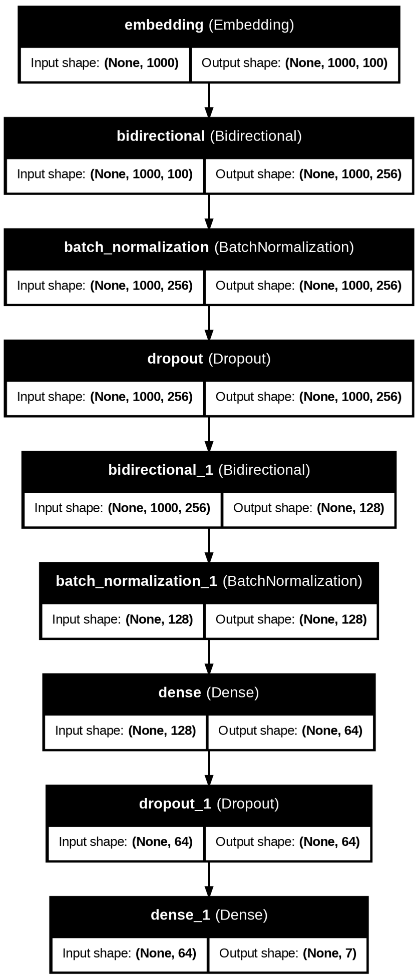

embedding (Embedding)
Input shape: (None, 1000)
Output shape: (None, 1000, 100)
bidirectional (Bidirectional)
Input shape: (None, 1000, 100)
Output shape: (None, 1000, 256)
batch_normalization (BatchNormalization)
Input shape: (None, 1000, 256)
Output shape: (None, 1000, 256)
dropout (Dropout)
Input shape: (None, 1000, 256)
Output shape: (None, 1000, 256)
bidirectional_1 (Bidirectional)
Input shape: (None, 1000, 256)
Output shape: (None, 128)
batch_normalization_1 (BatchNormalization)
Input shape: (None, 128)
Output shape: (None, 128)
dense (Dense)
Input shape: (None, 128)
Output shape: (None, 64)
dropout_1 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
dense_1 (Dense)
Input shape: (None, 64)
Output shape: (None, 7)

Figure 1: model architecture

We are using similar model architecture as we used in binary classifier case just in last layer we added 7 neurons instead of 1 as here we are doing multi-class classification and have 7 different classes.

We are getting accuraccy around 80% on validation data.

Chapter 5

Results and Discussion

We are plotting graphs of accuracy and loss while training the binary and multi-class classifiers.

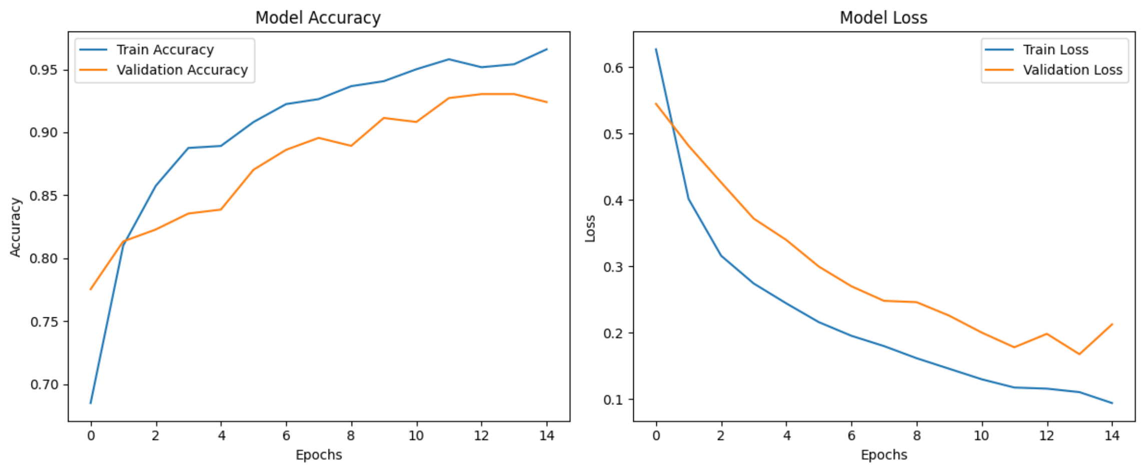


Figure 3: epochs vs accuracy and epochs vs loss for binary classifier

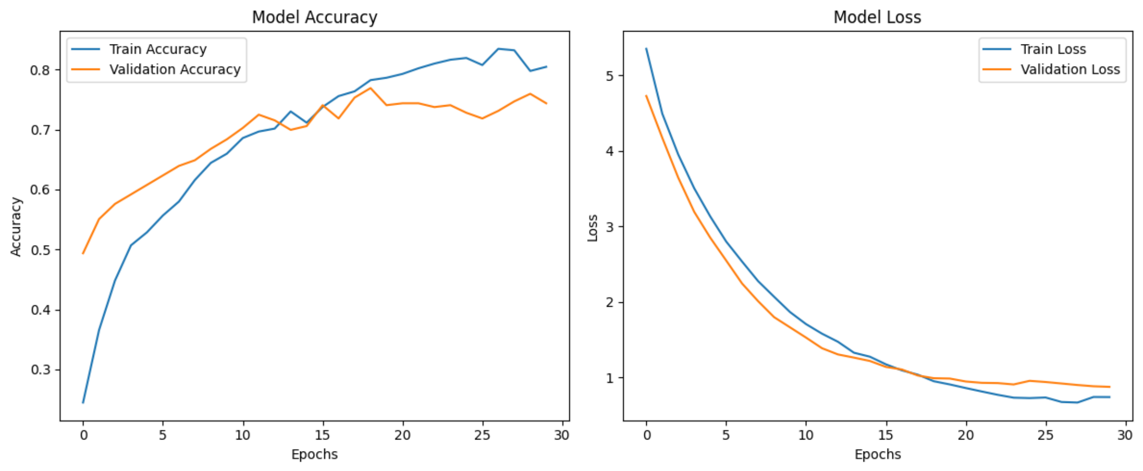


Figure 4: epochs vs accuracy and epochs vs loss for multi-class classifier

In [2], they evaluated many machine learning algorithms on the original dataset and their Modified Vector Space Representation technique on the ADFA-LD dataset.

Algorithm	**FP Rate**	**Precision**	**Recall**	**Accuracy (%)**
Naive Bayes	0.083	0.902	0.699	69.94
SMO	0.765	0.863	0.883	88.32
LibSVM	0.844	0.885	0.879	87.87
IBk (k=1)	0.124	0.96	0.96	95.95
kMeans (k=2)	0.858	0.749	0.722	70.64
ZeroR	0.875	0.765	0.875	87.46
OneR (B=5)	0.759	0.833	0.871	87.15
JRip	0.168	0.957	0.958	95.82
J48	0.154	0.959	0.96	95.98

Table 5.1: Performance Results for Binary Classification on the ADFA-LD Dataset

Algorithm	**FP Rate**	**Precision**	**Recall**	**Accuracy (%)**
Naive Bayes	0.028	0.885	0.588	58.814
SMO	0.845	0.819	0.877	87.683
LibSVM	0.850	0.796	0.876	87.565
IBk (k=1)	0.130	0.924	0.922	92.186
IBk (k=2)	0.192	0.919	0.924	92.405

Table 5.2: Performance Results for Multiclass Classification on the ADFA-LD Dataset

LSTM-based models give us decent results, but the models that they used in [2] are giving way better results than LSTM.

Chapter 6

Conclusion

We built intrusion detection system (IDS) using LSTM based classifiers(binary and multi-class) on ADFA-LD dataset [1] and got the accuracy of around 80-83% by both the models.We discovered that using Bi-directional LSTM is better than using the regular LSTM and LSTM may not be that efficient to build the intrusion detection systems other machine learning algorithms perform better [2] than LSTM.

Bibliography

[1] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 4487–4492, 2013.

[2] B. Borisaniya and D. Patel, "Evaluation of modified vector space representation using adfa-ld and adfa-wd datasets," *Journal of Information Security*, no. 6, pp. 250–264, 2015.

[3] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns." *Computers, IEEE Transactions on*, p. (99):11, 2013.

[4] G. Creech, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," 2014.