TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

PROJECT NO.: THA079MSISE01

USER BEHAVIOR ANALYTICS FOR INSIDER THREAT USING TRANSFORMER BASED APPROACH

BY

AARATI KUMARI MAHATO

SUPERVISED BY

ER. ROSHAN POKHREL

A MID TERM PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATICS AND INTELLIGENT SYSTEMS ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KATHMANDU, NEPAL

AUGUST, 2024

USER BEHAVIOR ANALYTICS FOR INSIDER THREAT USING TRANSFORMER BASED APPROACH

By:

Aarati Kumari Mahato

THA079MSISE01

Project Supervisor:

Er. Roshan Pokhrel

A project report submitted in partial fulfillment of the requirements for the degree of Master of Science in Informatics and Intelligent Systems Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Thapathali Campus

Tribhuvan University

Kathmandu, Nepal

August, 2024

ACKNOWLEDGMENT

Without the direction and aid of numerous people who in one way or another contributed and extended their invaluable assistance in the preparation and completion of this study, this project would not be feasible.
First and foremost, I would like to sincerely thank **Er. Roshan Pokhrel**, of Logpoint Nepal, my supervisor, for his essential advice, perceptive remarks, thorough ideas, and support over this project's duration. My heartfelt gratitude also goes out to **Er. Dinesh Baniya Kshatri**, the M.Sc. coordinator, for his tireless patience, insightful critique, and project management skills.
I am appreciative of Er. Om Prakash Mahato's direction, advice, and technical assistance with the identification, documentation, and investigation of project work.

Aarati Kumari Mahato
THA079MSISE01
August, 2024

ABSTRACT

One of the hot topics in cybersecurity right now is user behaviour analytics. In the past, attacks stemming from the deliberate or inadvertent actions of employees within the organisation did not typically worry people. Due to the frequent reports of data breaches involving many organisations and their own employees, businesses are growing increasingly concerned about the need to keep an eye on users' behaviour on the network. In light of this, the project work suggests a method for user behaviour analytics. This project makes use of the CERT (version 4.2) insider threat dataset, a five-log file synthetic dataset developed for insider threat research projects. Various raw events are provided by these log files. The behaviour analysis of every user in this project is carried out based on log sources. The structural behavior provided by CERT, is in structural format, which is converted into the contextual data bu fine-tuning the Llama3.1 model. These contextual data along with the synthesized contextual data are used to train the SecureBERT model for classification purpose. The model train accuracy was observed to start from 67% when trained and test accuracy was obtained to be 76% with 86% precision on 2500 data samples. The model was then trained on over 16 thousands data, observing the accuracy started with 92% in first epoch and reached to nearly 99.99%, which shows large language models can perform better and accurate in contextual user behavior analysis in insider threat.

Keywords: *Anomaly Detection,CERT,context generation, insider threat classification,Large language model, LLama, SecureBERT,Transformer*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBR	ABBREVIATIONS
APT	Advanced persistent threats
BERT	Bidirectional Encoder Representations from Transformers
CERT	Community Emergency Response Team
LLM	Large Language Model
Llama	Large Language Model Meta AI
ML	Machine Learning
NL	Natural Language
NLP	Natural Language Processing
RNN	Recurrent Neural Network
Roberta	Robustly Optimized BERT Pre-training Approach
SecureBERT	Secure Bidirectional Encoder Representations from Transformers
UBA	User Behavior Analytics
UEBA	User and Entity Behavior Analytics

1 INTRODUCTION

1.1 Background

As the digital environment continues to change and grow, cybersecurity has become a major concern for organizations all over the world. The rising dependence on digital systems and internet has increased the possibility of cybercrimes including malware, ransomware, or even phishing attacks and sophisticated intrusions such as APTs, which are short for Advanced Persistent Threats. In most cases these threats come from outside sources like hackers, criminal gangs involved in cyber crimes or government-sponsored groups who exploit vulnerabilities of software programs, hardware devices as well as human conduct with an aim of getting illegal access to sensitive data and systems. Financial losses, loss of customer confidence or reputation crises have forced companies to invest in information security that is firewalls, intrusion detection systems (IDS), Cryptography and comprehensive security policies.

But while the focus is typically on various external security dangers, insider threats present a somewhat trickier issue. Insider threats involves working for the organization and can be current, former employees, contractors or business partners that have legitimate access to your systems/data. Insiders can deliberately or inadvertently inflict harm through the disclosure of confidential information, disturbance in operations and carrying out external attacks. The problem with insider threats is that the perpetrators are trusted parties who have legitimate access, so their criminal activities can be more difficult to identify and stop.

Over the years, the problem of dealing with insider threats has been escalating because of the following reasons. New work models and individuals' ownership of at least part of their work-related activity on non-work devices also contributed to the increase in the attack surface that can be difficult to manage effectively. According to the 2023 Insider Threat Report by Cybersecurity Insiders, 72% of organizations have reported experiencing insider attacks, an increase from previous years. The report also highlights that the average annual cost of insider threats has risen to $15.4 million, underscoring the significant financial impact on businesses.

Also, organizations produce massive amounts of data that make it hard to distinguish strange activity that might point to an insider threat. The traditional security mechanisms are usually inadequate in identifying small changes that insiders might make that an

environment eludes or bypasses security measures. The same report goes to say that 60 percent of the organizations struggle to identify insider threats because of the challenges in differentiating between standard and suspicious behaviour. This stresses the need for complex approaches like machine learning and behavioral analysis to understand patterns and anomalies sufficiently.
Moreover, there is a large amount of data produced by organizations, which creates difficulties when searching for signs of improper behavior of insiders. Most of the traditional security approaches may not adequately detect regular, contextually appropriate activities that insiders may execute in a manner that violates security policies. According to the same report, it has revealed that 60% of the organizations struggle to identify the insider threats saying that it is hard to differentiate between the right behavior and the wrong one. This underlines the importance of the use of progressive approaches like the machine learning and the behavioral analytics in detecting patterns and anomalous behaviour.
Moreover, incentives of the insiders can be of different nature – from gaining personal material benefits, through resentment and desire for revenge to political beliefs, forced actions or blackmails. Due to this variability potential threats can be hard to predict and prevent appropriately. Insider Threat Report also explains that, 43% of the Insider Threat happen due to carelessness while 25% happen due to intention to harm, which also shows that the Insider Threat can be of various types. Consequently, the insider threat detection is still a research and practical field that is relatively important and constantly rising, and insiders are continually adapting and developing novel ways of protecting organizational resources and ensuring security.

1.2 Motivation

It is also fundamental, observing how in parallel to the digitalization of their assets and the growth of the availability of information, the nature and impact of threats have changed. Managers within business firms are equally powerful in a way they can misuse the resources of the firm and the employees opposed to Tom's proposal. They know the locations of valuable assets and its importance. Insiders through their authorized entry into an organization have the abilities to threaten the confidentiality, availability, or integrity of data. The insider threat can be considered as one of the biggest issues of today's cybersecurity as it has been acknowledged to be a significant threat that needs specialized approaches, tools, and mechanisms for reliable and accurate detection of the malicious insiders.

An insider who has ill uptake can for example bring down a key IT system, or download organization's trade secrets with the aim of assisting his or her new employer or rival company. Because the key missions of organizations heavily rely upon the usage of cyberspace, insider threat identification, when performed at the technology level, has become an intricate process that demands better methods and software integrated with machine learning systems to monitor potential indications of an attack and potential internal system loopholes. As it receives all the information in relation to the case, Insider Threat Detection allows an organization to adapt all its systems and practices to prevent the instances and dangers of insider threats and safeguard its valuable information.

The sources of insider threats also involve cyber observables which are actions that can be recorded on a computer. For instance, higher frequency of using web-based e-mail or cloud storage applications might imply an actual plan of carrying out an unauthorized replication of the Intellectual Property, thus, offering evident signs of a cyber attack. Other cyber observables are login and logout times, the files that have been opened in the network, access to the buildings via a card, and usage of e-mails in the organization. In the case of potential level one and when coupled with behavioral clues, such as poor or unpredictable performance at work, or complaining over missed promotion, it is possible to outline a pattern of a probable toxic employee.

The Insider Threat Detection project is aimed at increasing the organizations' monitoring of such observable activities both in the cyberspace and in the physical environment so that the occurrence and/or effects of insider threats can be prevented. Studying these indicators enables organisations to apply preventive measures to safeguard their important data and guarantee the reliability of their processes.

1.3 Problem Definition

Modern firewalls, antivirus programs, intrusion detection systems, and the like can identify active attacks like a sudden brute force attack, but they are largely unable to monitor passive user behaviour within the network. The majority of attackers carry out specific malevolent acts while seated beneath the rule. For instance, a person within an organisation may learn the privilege account authentication credentials and use them to steal and misuse sensitive data. We must observe user behaviour over time in order to

identify such actions. Numerous security and data breaches have been caused by users inside organisations. These items indicate the need for network-wide user behaviour monitoring.
Numerous studies conducted in this field have applied various algorithms using statistical methods for forecasting. But each user follows a certain pattern when it comes to doing their tasks. The suggested research project examines behavioural patterns or event sequences to identify aberrant behaviour. We must prepare the feature vector, which is the algorithm's input, before putting Large Language Model methods into practice. However, using a set time window for the feature's vector preparation may cause abnormal patterns to be missed. Anomalous patterns can split in a short time span. Extended duration is too generic. One option would be to capture the user's session rather than a set time range.

1.4 Project Objectives

The objective of the proposed project work are as follows:

- To filter the raw log events for each user and generate natural language contextual data (events) using large language model (llama3.1).
- To classify the user behavior(contextual events) between normal, scenario 1, scenario 2 and scenario 3 using SecureBERT model.

1.5 Scope of Project

Motivated from an industrial and organizational needs, a Transformer-based user behavior detection and is developed which learns the CERT dataset as well as the augmented data which replaces the email content present in email log file of CERT dataset by the self-generated email content by using the large language model dataset in a semi-supervised way in the training process. The transformer network model is large language model which is very popular in these days will be used to improve the performance of the machine learning landscape of UEBA.

1.6 Potential Project Applications

User Behavior Analytics applications can be used to identify between the normal and abnormal behavior of the user through the analysis of previous activities performed by the particular user compared to a known reference imprint (in some cases, of the user, in other cases, of a user profile) or detect malicious or abnormal behavior of a legitimate user. By analyzing patterns of user behavior, organizations can gain insights that drive a wide range of applications across different industries. It can be applicable in:

- Threat detection
- Fraud prevention
- Incidence Response

1.7 Originality of Project

Developing this project, contribution will as follow:

- Transform the raw log events of CERT4.2 dataset into Natural Language context
- Fine-tune the Llama3.1 model for natural language contextual data generation
- Implement secureBERT model for the classification between normal and malicious instances

1.8 Organization of Project Report

This project report's content is arranged into seven chapters. Following the introduction and discussion of the problem topic in chapter 1, chapter 2 presents a survey of important and pertinent literature that highlights a significant research gap. The project's implementation and methods are covered in Chapter 3. The results are presented in Chapter 4, and Chapter 5's analysis and discussion come next. The future enhancements are listed in Chapter 6, and the conclusion of the project are discussed in Chapter 7 which is followed by the appendices and references.

2 LITERATURE REVIEW

Organisations must deal with financial loss and information leakage as a result. Employers are facing an increasing problem in the form of insider threats, which are complicated since attackers may act like regular individuals. Any activity carried out by an employee that could be detrimental to the company, such as unauthorised data transfers, resource sabotage, or unauthorised access to resources, is typically referred to as an insider attack. An employee's dissatisfaction can take many different shapes when it comes to insider threats, from advanced persistent threats (APT) that plan multi-year campaigns to obtain and retrieve intelligence data to a dissatisfied employee undermining the reputation of an organisation.. [7]

By monitoring only the top 1 percent of suspicious cases, the anomaly detection model produced at most 53.67 percent of the detection rate based on the daily activity summary information. For two of the three jobs under evaluation, more than 90% of genuine anomalous behaviours were found when the monitoring coverage was expanded to include the top 30% of anomaly scores. When the top 30 percent of suspicious emails were monitored, the detection rate jumped to 65.64 percent (98.67 percent at most) compared to the maximum of 37.56 percent for malicious emails identified using the 1 percent cut-off value in the email content dataset evaluation. Despite the fact that the suggested paradigm has empirical backing, the present study has certain drawbacks. There is a lack of proper integration between the various anomaly detection outcomes and this approach cannot detect malicious behavior in real time.[3]

A 2018 evaluation of the insider threat found that 53% of attacks originated from within organisations in the preceding year. Furthermore, according to 27% of the companies surveyed, the attacks originated within. It was shown that the likelihood of insider threats is roughly 66% when there is network access and 27% when there is physical access. A taxonomy of modern insider types, access, level, motivation, insider profiling, effect security property, and techniques used by attackers to conduct attacks were presented, along with an overview of noteworthy recent works on insider threat detection that covered the analysed behaviours, machine-learning techniques, dataset, detection methodology, and evaluation metrics. They also introduced CERT, NSL KDD OR KDD-99, APEX, RUU, and TWOS dataset.[1]

The author [6] presented the UEBA that uses predefined Rules, Anomaly Detection and Machine Learning techniques in its process of hunting insider attacks. According to Insider Threat Report 2019, 68 percent of organizations are experiencing frequent insider attacks. Data containing valuable information regarding user activities for example, login and logout information, email activities, web searches, files activities, servers accessed, applications, USB etc. is collected from various sources such as system logs, application logs, network devices, network traffic, net flows etc. In this paper They discussed the UEBA techniques put out in the literature, the broad design and features of the best UEBA solutions now on the market, as well as their advantages and disadvantages. There were several issues that needed to be addressed in development and deployment of the UEBA system which were duration of training data, reducing false positives, risk score calculation. In evaluating UEBA solutions, threat detection capabilities must be taken into account in addition to use cases and scalability. Advanced behavior analytics system architecture that incorporates information from many data sources including social media activities, email and network access logs are also required.

This work, [5], uses the RNN (LSTM) architecture to learn the behavioural pattern of logs and compute their anomaly scores for time-series evaluation. It makes use of kafka. After going through a Kafka topic, the combined logs from Logstash are given to the best deep learning model. Because RNN-based techniques can handle temporal log data, they were deemed to be the best option. CNN has proven to be yet another effective approach when paired with RNN. After being routed through a Kafka topic, these outcomes are indexed into Elasticsearch for additional examination. On the Kibana dashboard, this stored data is shown in real-time as graphs. They have made an effort to provide readers with the necessary information in this survey study so they can begin the challenge of locating a needle in the haystack that is insider threat detection. They have talked about the importance of detecting insider threats as well as the combination of log-based anomaly detection with deep learning. The creation and gathering of insider threat data may occupy a whole scientific domain. In this context, few-shot learning-based techniques can be applied. While there are fully or partially developed answers for some of these problems, the remaining ones are still being investigated.

The authors of the paper [4] came to the conclusion that specialised datasets for particular domains can now be fine-tuned with pre-trained models on bigger datasets thanks to transfer learning-based techniques. The most recent developments in natural language processing (NLP) models can help cybersecurity operations by facilitating the proactive identification of different threats. Using a pre-trained language model named MalBERTv2, they provide a novel method in this study for capturing the relevance and significance of the Malware/Goodware (MG) datasets. The MalBERTv2 classifier's BERT layer block features a fully connected layer for the prediction probabilities in addition to having the same weights as the pre-trained BERT. We introduce MalBERTv2, an end-to-end malware/goodware language paradigm for malware categorisation, combining all these components.

The authors of the paper [4] concluded that transfer learning-based techniques have made it possible to fine-tune specialised datasets for certain domains with pre-trained models on larger datasets. Cybersecurity operations can benefit from the latest advancements in natural language processing (NLP) models, which can aid in the proactive detection of various threats. In this work, they present a novel approach for capturing the relevance and significance of the malware/goodware (MG) datasets using a pre-trained language model called MalBERTv2. The BERT layer block in the MalBERTv2 classifier has the same weights as the pre-trained BERT, and it also has a fully connected layer for the prediction probabilities. Bringing all these elements together, we present MalBERTv2, an end-to-end malware/goodware language paradigm for malware classification.They use a classifier as a stage in the model pipeline that utilises bidirectional encoder representations from transformers (BERT). The performance of our model is evaluated on multiple datasets, and a weighted f1 score in the range of 82% to 99% is obtained. The lack of benchmarks for malware and software identification was one of the obstacles that prevented them from conducting an effective comparison between the model and existing methods.

LM-Hunter is a novel approach for large-scale APT adversary hunting that makes use of graphs and NLP technologies. LM-Hunter processes subgraphs of user login activity using seq2seq models to find the most questionable lateral movements in the network. LM-Hunter offers a comprehensive perspective of the user's lateral motions within the network, in contrast to UEBA solutions. The contextualisation of user activity is made easier by the use of graphs, which helps cybersecurity teams spot stealth APT actors cruising the network. Following a comparison of ten executions, there was very little variation in the model outputs. The models that were trained between 65% and 85% accuracy rates produced the greatest outcomes. [3]

LM-Hunter is a novel approach for large-scale APT adversary hunting that makes use of graphs and NLP technologies. LM-Hunter processes subgraphs of user login activity using seq2seq models to find the most questionable lateral movements in the network. LM-Hunter offers a comprehensive perspective of the user's lateral motions within the network, in contrast to UEBA solutions. The contextualisation of user activity is made easier by the use of graphs, which helps cybersecurity teams spot stealth APT actors cruising the network. Following a comparison of ten executions, there was very little variation in the model outputs. The models that were trained between 65% and 85% accuracy rates produced the greatest outcomes. Additionally, the language models showed promise in identifying the purpose of phishing emails, occasionally beating people in identifying malicious intent in less evident phishing emails. There were notable differences in the performance and stability of the four examined models (GPT-4, Claude, Bard, and LLaMA), with Claude offering the most reliable and practical outcomes. In addition, Claude offered helpful advice on how to respond to phishing emails, like visiting the business's official website to confirm any gift card offers. The capabilities of LLMs are being researched at a rapid pace, and findings become dated soon. Rather than serving as a conclusion, the trials presented in this article should be viewed as a starting point for more research. Their research goal for the future looks into how LLM-based, customised phishing training may be used in the real world. [2]

3 METHODOLOGY

3.1 Theoretical Formulations

In this project, we use a large language model by meta AI (Llama3.1) for the contextual data generation from the raw log events dataset i.e. the CERT4.2 dataset. The Secure-BERT model is used for the classification purpose between the normal and malicious behaviors of a user.

LLAMA Model for Contextual Data Generation:

Large Language Model Meta AI (LLAMA) is a more advanced and highly capable transformer-based model to produce semantically sound text. In this particular project, the raw CERT4.2 data have been analyzed according to the identified LLAMA model that practices change throughout the process. The dataset entries are converted into natural language textual descriptions related to the users' activities. Thus, these descriptions give a clear picture on the user activities making it easier to analyze and deduce patterns that may be attributed to insiders.

Pre-Processing Steps: Eliminate the redundancy, dealing with absence of data as well as standardization of format of data. Grouping of the records belonging to the same user activity into a single record that contains information on all these activities is done. These data are then tokenized into tokens that are digestible by the LLAMA. Then, contextual embedding is applied to the tokens to capture the context of the meaning of the user activities.

Post-Processing Steps: Creation of the descriptive texts from the embedded user activities, applying the LLAMA model is done. Then, text normalization is done, making sure that the developed texts are properly articulated and semantically accurate in the given context. Finally the generated texts are assigned to the corresponding specific categories such as normal or malicious according to certain previously defined rules.

SecureBERT for Classification:

More specifically, secureBERT is a version of BERT, that is fine-tuned for tasks in the security domain only. For this purpose in the current project, SecureBERT is employed for categorizing the textual descriptions into normal and malicious behaviors.

Pre-Processing Steps: Regarding textual descriptions, like in the case of LLAMA, the descriptions are preprocessed for SecureBERT by tokenizing it.The tokens are then transformed into embedding vectors with the help of BERT pre-trained embeddings.

Post-Processing Steps: SecureBERT denotes an input as normal or malicious if its embedded vectors belong to the former region of the PCA plot or the latter region respectively.

Major Benefits of the Chosen Technique:

The LLAMA model produces specific and contextually enriched information from the logs and gives a better picture of the user behavior. It is also worthy of note that the transformer-based solution can model complex patterns and relationships of insider threats in penetrating depth and comprehensive detection. Also, both algorithms are highly scalable, thus they can be used for large data sets and are perfect for the enterprise-level UBA. The models can be further trained in real-time mode that will allow identifying and responding to insider threats in a timely manner.

For categorization and user behaviour analytics activities, Large language models are quite beneficial. There are many important advantages to using LLaMA 3 for insider threat categorization and detection on the CERT dataset. The CERT dataset is a great tool for training and assessing machine learning models in cybersecurity contexts since it contains a variety of behavioural data, including network activity, email exchanges, and user actions. The CERT dataset is intended to mimic real-world insider threat scenarios. Because of its sophisticated natural language processing capabilities, which enable it to comprehend and evaluate the complex textual data frequently involved in insider threats, LLaMA 3 performs exceptionally well in this application. Because of the model's skill in contextual analysis, even when seemingly benign behaviours disguise small behavioural anomalies and patterns indicative of possible insider threats, they can be identified.

Moreover, LLaMA 3 can be precisely tailored to the subtleties of the CERT dataset thanks to its transfer learning and fine-tuning capabilities, which increases its efficacy in identifying and categorising various insider threat categories. Through the utilisation of the CERT dataset's intricate simulations of diverse insider threat scenarios, LLaMA 3 is able to cultivate an advanced comprehension of diverse threat vectors, ranging from fraud and espionage to data exfiltration and sabotage. Because of its versatility, LLaMA 3 offers a proactive approach to threat detection by enabling it to recognise both established and new threat patterns.

Furthermore, by handling unstructured and semi-structured data, the model enables thorough analysis across various data types, guaranteeing that no important information is missed. Large datasets can be handled by LLaMA 3 with ease, and its scalability and integration capabilities allow it to function with current security frameworks to deliver reliable, real-time threat detection. Because of its sophisticated features, security teams may concentrate on the biggest dangers by using it as a potent tool for threat identification and prioritisation. Overall, LLaMA 3's rich contextual knowledge, processing versatility, and fine-tuning flexibility make it an effective tool for proactive insider threat management that can handle challenging security situations in dynamic settings.

Assumptions taken into account:

Concerning the raw CERT 4.2, the analysis assumes rocking high quality data, where the quantity of noise and errors is relatively low, therefore, maximum effective outcome is received. The first and very important step of the big data analysis is data pre-processing; this data must be cleaned before they can be fed into the model, otherwise they cannot be admitted. Both, the employed LLAMA for contextual data generation and the SecureBERT for classification are pre trained on huge and diverse data. This large amount of pre training should be generic and specific tuning on datasets should provide the particularity that is required for insider threat detection.

Moreover the method presupposes that the malicious behaviors are not similar to normal behaviors and there is likely a hood of the observation of the malicious behaviors using the context analysis and classification. It is also assumed that there is ample GPU or TPU to train and deploy these transformer models. Also, there is a touted notion that the constructed models will be recurrent with fresh data from time to time because threats are dynamic and evolve, thus ensuring high capabilities and robustness.

Thus, including the contextual data generation using the LLAMA model and SecureBERT for the classification of the users' behavior, this transformer-based solution represents a more efficient and integrated approach to the insider threat identification in the user behavior analysis.

3.2 Mathematical Modelling

3.2.1 Neural Networks

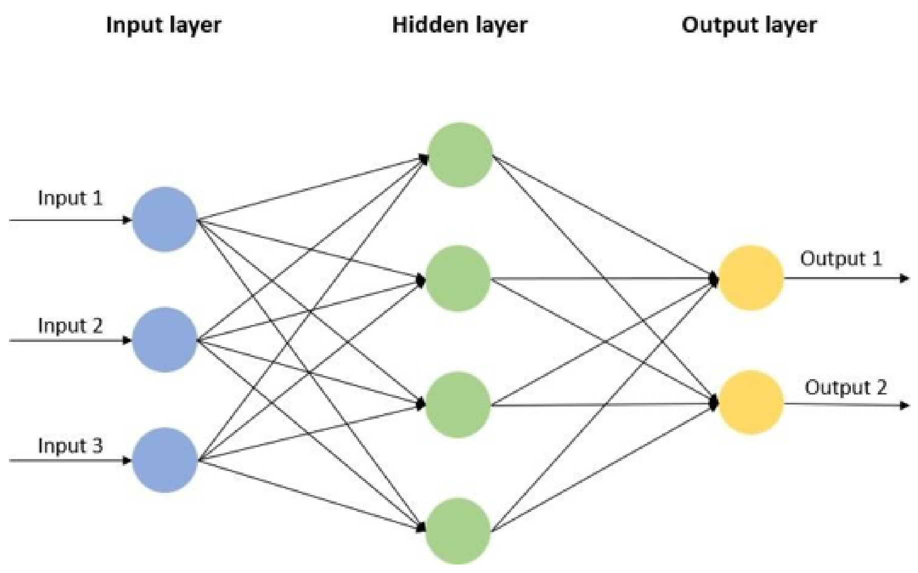


Figure 3.1: Simple fully-connected neural network architecture

LLMs are built on deep learning principles, utilizing neural networks, specifically transformer architectures. The primary goal is to create models that can understand the context and meaning of words within large text sequences, enabling them to generate coherent and contextually appropriate text.

3.2.2 Large Language Model

Large Language Models (LLMs) represent a significant advancement in the field of natural language processing (NLP). These models are capable of understanding and generating human-like text by leveraging vast amounts of data and complex neural network architectures. This document provides an in-depth explanation of LLMs, focusing on their core concepts, architectures, training methodologies, applications, and specific benefits. Additionally, we will delve into the mathematical foundations and illustrative figures that explain the inner workings of these models.

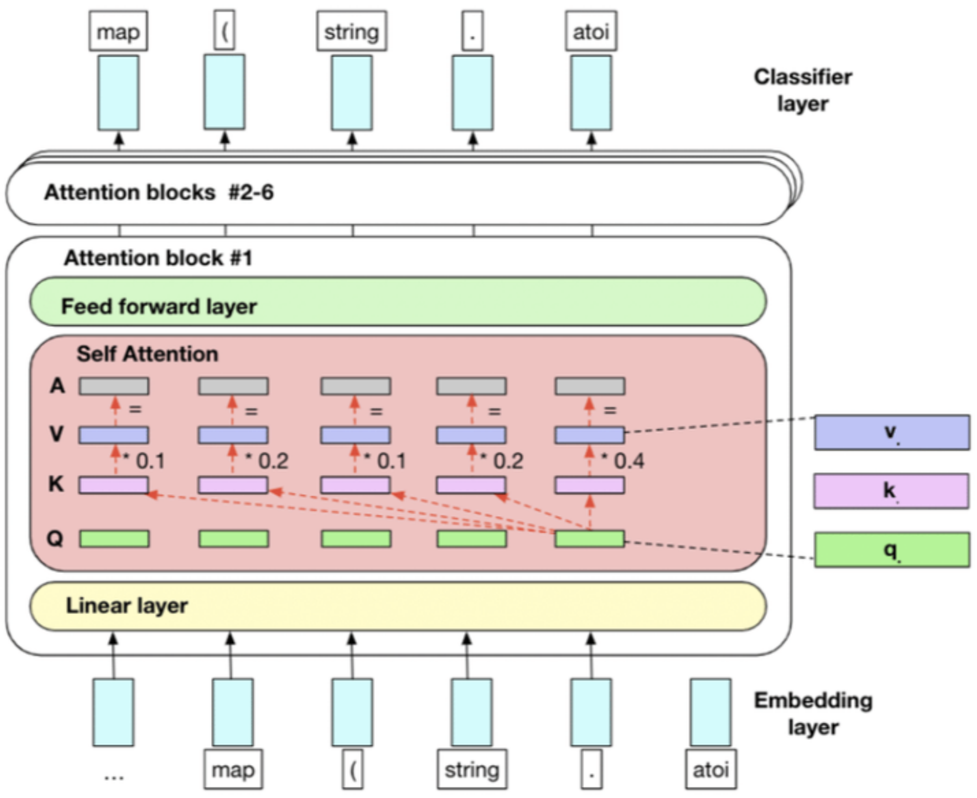


Figure 3.2: Architecture of LLM

Large Language Model by Meta AI(Llama 3.1

The third version of Meta's LLaMA series, LLaMA 3 (Large Language Model Meta AI), aims to improve natural language processing (NLP) capabilities. Although Meta's particular implementations would determine the precise architecture details for LLaMA 3, we can deduce broad concepts from earlier versions (such as LLaMA 2) and the most advanced transformer models. LLaMA models use a decoder-only architecture, focusing on generating text based on the input context.

Transformer Architecture: The Transformer architecture, introduced in the paper ”Attention Is All You Need,” by Vaswani et al. in 2017, has been a game-changer for NLP tasks and is the foundation of most modern LLMs. It replaces traditional recurrent and convolutional neural networks with self-attention mechanisms that allow for better handling of long-range dependencies in sequences, by allowing the model to weigh the significance of different words in a sentence when making predictions. The transformer architecture consists of an encoder and a decoder:

- Encoder: Processes the input text and generates a series of context-aware representations.
- Decoder: Uses these representations to generate the output text.

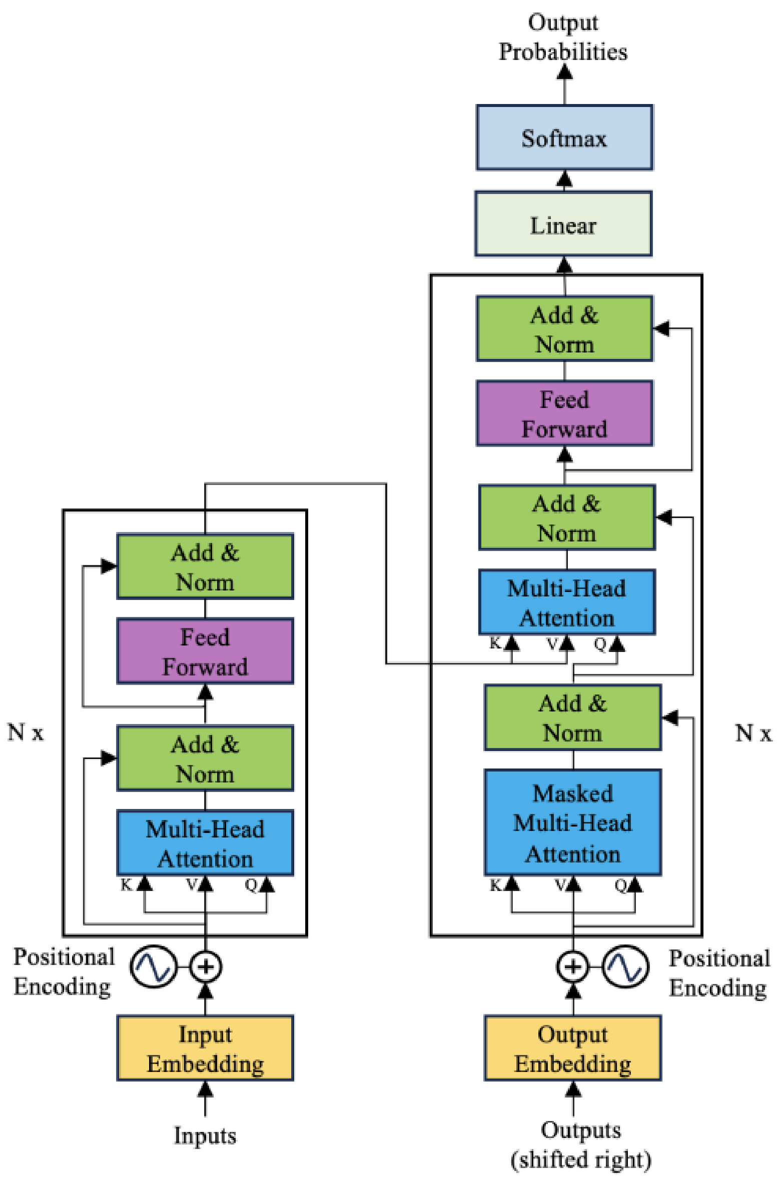


Figure 3.3: Transformer Architecture

Components of Transformer

- **Input Embedding:** Converts input words into dense vectors of fixed size.
- **Positional Encoding:** Adds information about the position of words in the sequence to the embeddings.
- **Multi-Head Self-Attention Mechanism:** Allows the model to focus on different

parts of the input sequence, capturing dependencies between words regardless of their distance from each other in the text.

- **Feed-Forward Neural Network:** Applies non-linear transformations to the attention outputs.
- **Add and Norm:** Adds the input and output of each sub-layer (residual connection) and normalizes it.

Self-Attention Mechanism The self-attention mechanism enables the model to focus on different parts of the input sequence when encoding a particular word. This mechanism helps capture the dependencies between words, making it crucial for understanding context.

The self-attention score α_{ij} for the i-th word in relation to the j-th word is computed as follows:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{3.1}$$

Where e_{ij} is the compatibility function defined as:

$$e_{ij} = \frac{(Q_i \cdot K_j^T)}{\sqrt{d_k}} \tag{3.2}$$

where,

- Q(Query), K(Key) and V(Value) are the projections of the input embeddings.
- d_k is the dimension of the key vectors.

Multi Head Attention

Multiple attention heads are employed to enable the model to jointly attend to data from various representation subspaces:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O \tag{3.3}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (3.4)$$

and W^O is the output projection.

The multi-head attention mechanism allows the model to capture different aspects of the relationships between words, leading to better understanding and generation of text.

Feed Forward Network

Each attention output is fed into a feed-forward neural network (FFN), which consists of two linear transformations with a ReLU activation in between. The output of this network is:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (3.5)$$

Where, W_1 and W_2 are weight matrices , and b_1 and b_2. This network applies non-linear transformations to the attention outputs, enhancing the model's ability to capture complex patterns in the data.

Positional Encoding

Since transformers do not have a built-in sense of the order of the words, positional encoding is added to the input embeddings to provide this information. This encoding helps the model understand the sequence of words, which is essential for tasks like language modeling and translation.

The positional encoding for each position *pos* and dimension *i* is defined as:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (3.6)$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (3.7)$$

where, d_model is the dimension of the model.

Training and Fine Tuning

LLMs are typically pre-trained on large corpora of text using unsupervised learning techniques, such as masked language modeling (MLM) or next-word prediction. This pre-training helps the model to learn grammar, facts, and some level of reasoning. After pre-training, LLMs can be fine-tuned on specific tasks or datasets to improve their performance in particular domains.

Pre-Training

Pre-training involves training the model on a large dataset to predict missing or next words in sentences. This phase helps the model learn the general structure and nuances of the language.

Masked Language Modeling (MLM) In MLM, random words in a sentence are masked, and the model is trained to predict these masked words based on their context. This approach helps the model understand the context and relationships between words.
For example, in the sentence ”The quick brown [MASK] jumps over the lazy dog,” the model should predict the masked word ”fox.”

Next-Word Prediction In next-word prediction, the model is trained to predict the next word in a sentence given the preceding words. For example, given the input ”The quick brown fox jumps,” the model should predict ”over.”

Fine-Tuning Fine-tuning involves training the pre-trained model on a smaller, task-specific dataset to adapt it to particular applications. This step enhances the model’s performance on specific tasks, such as sentiment analysis, question answering, or insider threat detection. Some of the fine tuning techniques for large language models are:

- **Task-Specific Head Training:** LLaMA 3, similar to other transformer-based models, typically fine-tunes the last few layers or adds task-specific heads on top of the existing architecture. These heads are trained on the downstream task data to specialize the model’s outputs for the specific task. For example, for the purpose of a classification task, a linear layer with softmax activation can be added on top of the model’s output to classify the input data into different categories.

  Let h_L represent the outputs from the last layer of the transformer encoder. For a classification task with C classes, the classification head can be defined as:

$$\text{softmax}(W_{\text{class}}\mathbf{h}_L + \mathbf{b}_{\text{class}}) \tag{3.8}$$

Where,

- $\mathbf{h}_L$ is the output from the last layer of the language model,
- W_{class} is the weight matrix for the classification layer,
- $\mathbf{b}_{\text{class}}$ is the bias vector for the classification layer.

- **Gradient Descent and Backpropagation:** During fine-tuning, the model parameters are updated using gradient descent optimization techniques. The gradients are computed using backpropagation, where the loss function for the task is differentiated with respect to the model parameters.
  The parameters θ of the model are updated iteratively using the gradient descent update rule:

  $$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathscr{L} \tag{3.9}$$

  where:

  - η is the learning rate, determining the step size of each iteration,
  - $\mathscr{L}$ is the loss function, representing the discrepancy between predicted and actual values.

- **LoRA (Local Representation Alignment):**LoRA aims to align the representations between the pre-trained model and task-specific data by minimizing the discrepancy between their distributions. This is achieved through local alignment operations during fine-tuning.
  Objective function: The objective of LoRA involves minimizing the discrepancy between the pre-trained representations $\mathbf{h}_i^{\text{pre}}$ and the task-specific representations $\mathbf{h}_i^{\text{task}}$ across a batch of examples. This is typically done using Mean Squared Error

(MSE) between these representations:

$$\mathscr{L}_{\text{LoRA}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{h}_i^{\text{pre}} - \mathbf{h}_i^{\text{task}}\right\|^2 \tag{3.10}$$

where,

- N is the batch size.
- $\mathbf{h}_i^{\text{pre}}$ is the pre-trained representation for example i.
- $\mathbf{h}_i^{\text{task}}$ is the task-specific representation for example i.

Local Alignment Operation To perform local alignment, LoRA computes task-specific representations $\mathbf{h}_i^{\text{task}}$ using an alignment function f_{align} that adjusts $\mathbf{h}_i^{\text{pre}}$ based on task-specific data features $\mathbf{x}_i$:

$$\mathbf{h}_i^{\text{task}} = f_{\text{align}}(\mathbf{h}_i^{\text{pre}}, \mathbf{x}_i) \tag{3.11}$$

Gradient Descent Update: During fine-tuning with LoRA, the model parameters θ, which include both the pre-trained parameters and the parameters of the alignment function, are updated using gradient descent. The gradient with respect to θ for the LoRA loss $\mathscr{L}_{\text{LoRA}}$ is computed using backpropagation:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathscr{L}_{\text{LoRA}} \tag{3.12}$$

where, η is the learning rate, and $\nabla_\theta \mathscr{L}_{\text{LoRA}}$ denotes the gradient of the LoRA loss with respect to the model parameters θ.

3.2.3 Bidirectional Encoder Representations from Transformers(BERT)

BERT is a state-of-the-art language representation model designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. This allows BERT to understand the context of a word based on its surroundings, making it highly effective for a wide range of natural language processing tasks.BERT is built on the Transformer architecture, which uses self-attention mechanisms to process input text. The Transformer consists of an encoder and a decoder, but BERT only uses the encoder part. The input to BERT is tokenized text, which is converted into token embeddings, segment embeddings, and positional embeddings. These embeddings are then processed through multiple layers of self-attention and feed-forward neural networks.

The self-attention mechanism in BERT can be described with the following equations. Given an input sequence of tokens, the self-attention mechanism computes a weighted sum of the input representations.
For insider threat classification, BERT can be fine-tuned on a labeled dataset of insider activities.

1. Preprocessing: Tokenizing the text data from insider threat logs (e.g., logon times, file accesses, email content) and converting it into a format compatible with BERT.
2. Fine-Tuning: Training the BERT model on the labeled dataset where the labels indicate whether the activity is normal or malicious. The classification head, usually a simple feed-forward neural network, is added on top of BERT's output.
3. Prediction: Using the fine-tuned model to classify new data. The model outputs the probability of an activity being malicious.

3.2.4 SecureBERT

SecureBERT is a specialized language model designed for cybersecurity applications, particularly effective in understanding and classifying cybersecurity-related texts, including those related to insider threats. It was developed by fine-tuning the RoBERTa-base model with a substantial dataset comprising 98,411 cybersecurity-related textual elements (equivalent to about 1 billion tokens). This fine-tuning process involved introducing noise into the token weights during training to enhance robustness against adversarial attacks. The model aims to capture the nuances of cybersecurity language, making it particularly effective for tasks such as Cyber Threat Intelligence (CTI) analysis, especially in identifying and classifying insider threats.

Table 3.1: Key statistics of the dataset used for training SecureBERT

Type	**No. Documents**
Articles	8,955
Books	180
Survey Papers	515
Blogs/News	85,953
Wikipedia (cybersecurity)	2,156
Security Reports	518
Videos	134
Total	**98,411**

Table 3.2: Additional statistics of the dataset used for training SecureBERT

Metric	**Value**
Vocabulary size	1,674,434 words
Corpus size	1,072,798,637 words
Document size	2,174,621 documents (paragraphs)

Table 3.3: Sources of textual data collection for training SecureBERT

Category	**Sources/Tags**
Websites	Trendmicro, NakedSecurity, NIST, GovernmentCIO Media, CShub, Threatpost, Techopedia, Portswigger, Security Magazine, Sophos, Reddit, FireEye, SANS, Drizgroup, NETSCOUT, Imperva, DANIEL MIESSLER, Symantec, Kaspersky, PacketStorm, Microsoft, RedHat, Tripwire, Krebs on Security, SecurityFocus, CSO Online, InfoSec Institute, Enisa, MITRE
Security Reports and Whitepapers	APT Notes, VNote, CERT, Cisco Security Reports, Symantec Security Reports
Books, Articles, and Surveys	Tags: cybersecurity, vulnerability, cyber attack, hack ACM CCS: 2014-2020, IEEE NDSS (2016-2020), IEEE Oakland (1980-2020), IEEE Security and Privacy (1980-2020), Arxiv, Cybersecurity and Hacking books
Videos (YouTube)	Cybersecurity courses, tutorials, and conference presentations

SecureBERT accepts input sequences of variable lengths, typically up to 512 tokens, allowing it to process diverse and complex text data effectively. The model architecture comprises 12 hidden transformer layers, which facilitate deep contextual understanding through the self-attention mechanism inherent in transformer models. After processing through the transformer layers, the output is passed to a classification head, which

includes, a fully connected hidden layer, an output layer that produces class probabilities and softmax activation function to normalize the output generated into a probability distribution. With approximately 123 million parameters, it leverages significant computational power to capture intricate patterns and relationships in text, making it well-suited for tasks such as identifying and classifying insider threats.
Consider an insider threat scenario involving a departing employee at a tech company who is attempting to steal sensitive information, to illustrate the working of secureBERT.
Scenario: A departing employee at a tech firm downloads sensitive company documents, including proprietary software code and customer data, just days before leaving for a competitor. This action poses a significant risk to the company's intellectual property and customer privacy.

- **Tokenization:** The text describing the incident is tokenized into subword units, resulting in tokens such as [”departing”, ”employee”, ”downloads”, ”sensitive”, ”documents”, ”before”, ”leaving”, ”for”, ”competitor”].
- **Embedding:** Each token is converted into an embedding vector using the model's embedding layer, which captures the semantic definition of the words in the context of insider threats.
- **Transformer Processing:** The embeddings are processed through the 12 transformer layers. During this stage, SecureBERT uses self-attention mechanisms to identify relationships between tokens, such as the connection between ”departing employee” and ”downloads sensitive documents”. This allows the model to recognize the potential threat posed by the employee's actions.
- **Output Generation:** The final hidden state is passed through the classification head, which outputs probabilities for various classes, such as ”malicious”, or ”normal”. For this eg, the model might classify the incident as a ”malicious” threat due to the intentional nature of the data theft.

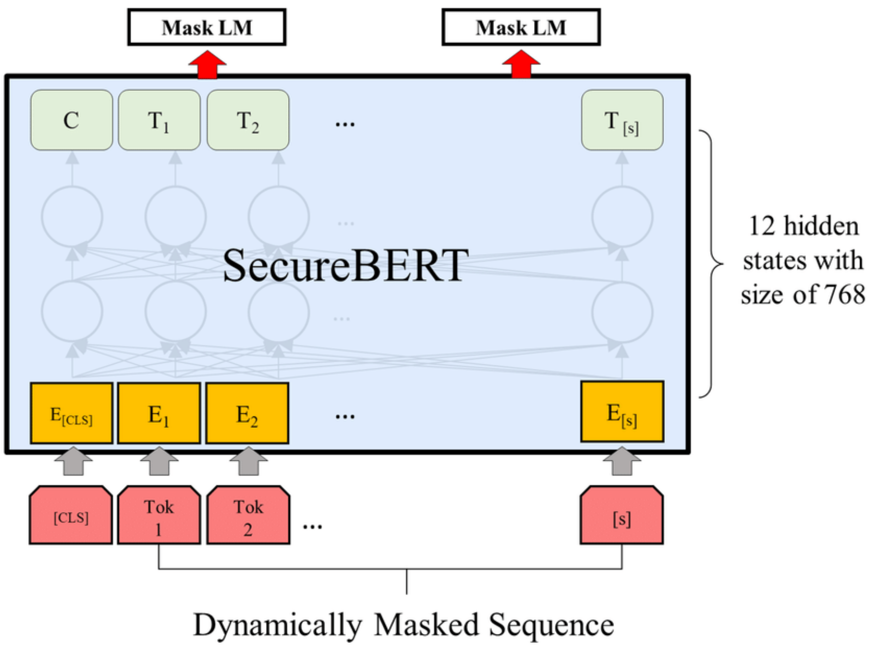


Figure 3.4: Architecture of SecureBERT

3.2.5 Leveraging LLMs with the CERT Dataset

The CERT dataset, designed to emulate real-world insider threat scenarios, includes diverse types of behavioral data, such as network activities, email communications, and user actions. This dataset provides a rich resource for training and evaluating LLMs in cybersecurity contexts.

LLMs, such as LLaMA 3, excel in this application due to their advanced natural language processing capabilities. They can analyze complex textual data, identify subtle behavioral anomalies, and detect patterns indicative of potential insider threats.

Benefits in Insider Threat Detection

- **Comprehensive Analysis:** LLMs can process and analyze various types of data from the CERT dataset, including logs, emails, and user actions, to detect potential threats.
- **Real-Time Detection:** The ability to perform real-time analysis enables proactive threat detection, allowing security teams to respond quickly to potential risks.
- **Adaptability:** LLMs can be fine-tuned on the CERT dataset to enhance their performance in recognizing specific insider threat scenarios.

- **Scalability:** The models can handle large volumes of data, making them suitable for deployment in environments with extensive security monitoring needs.

3.3 System Block Diagram

3.3.1 Training Phase

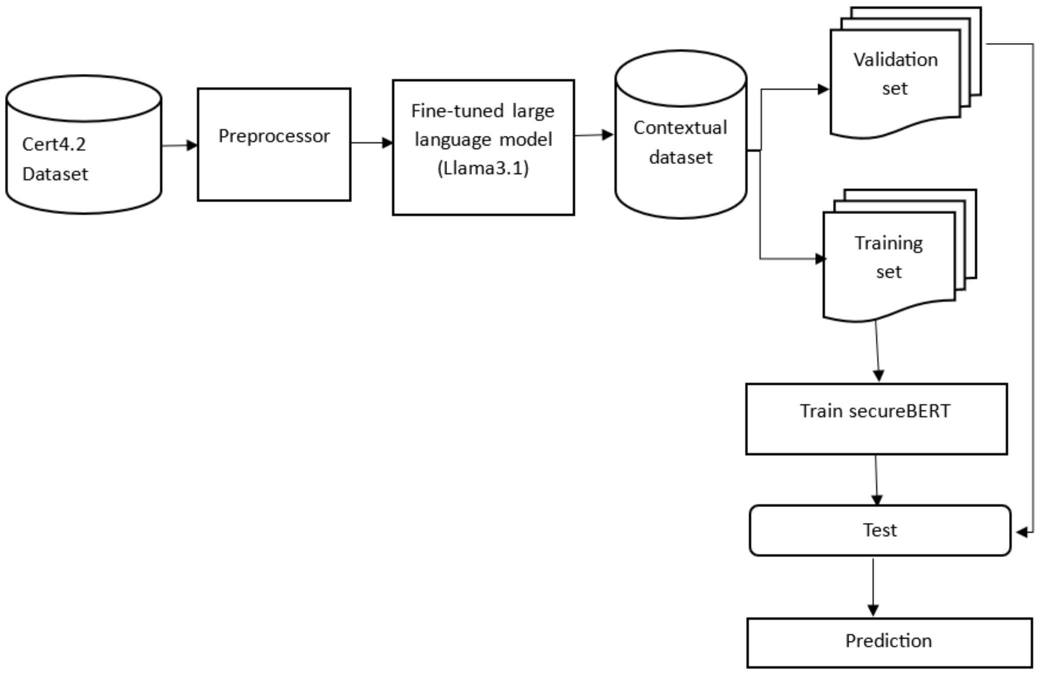


Figure 3.5: Block diagram Training phase of SecureBERT

The above figure shows the training phase of secureBERT model. The initial input for this phase the original log events data from CERT4.2 dataset. Which is then passed into the preprocessor. After data preprocessing, the preprocessed data is passed into the fine tuned Llama3.2 model for contextual data generation from the raw log events. The process how llama3.1 was fine tuned is explained later. Now, the contextual labeled data has been obtained by the llama model, which is then divided into the train data and validation data. Using the train set of data, secureBERT model is trained for the classification task by adding a classification head for the classification between normal and abnormal. This yields into a custom trained secureBERT model whiose architecture is explained later. The custom trained secureBERT model was trained on different combinations of learning rate, epochs and batch size. GridSearch Cross Validation technique was used to search and find the optimal combination of the hyperparameters. In first phase, the model was only trained for the normal and threat(abnormal) data.

Followed by the second phase of training, which was done for the threat data that is training on the three scenarios(type1, type2, type3).

3.3.2 Testing Phase

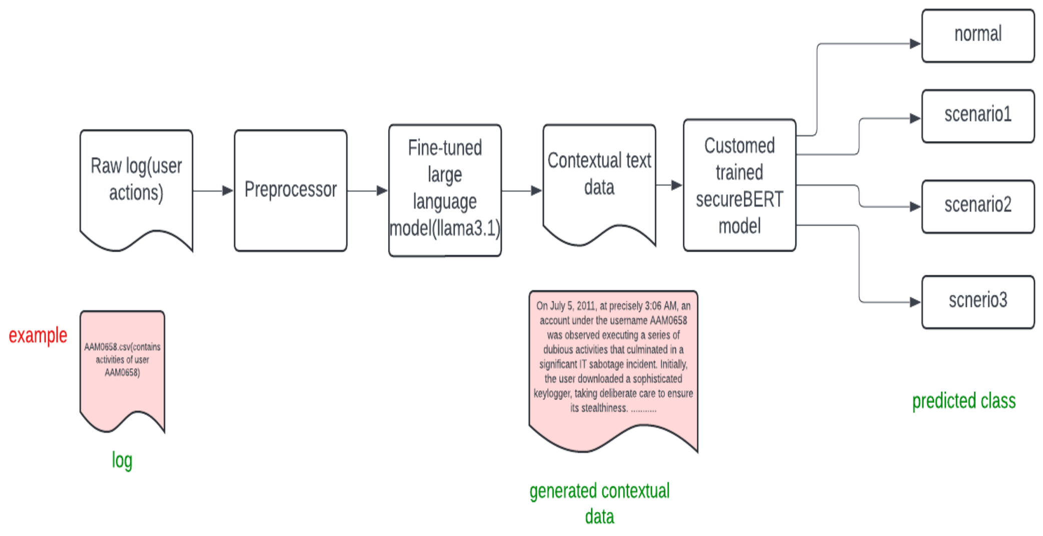

Raw log(user actions)
Preprocessor
Fine-tuned large language model(llama3.1)
Contextual text data
Customed trained secureBERT model
normal
scenario1
scenario2
scenio3
example
AAM0658.csv(contains activities of user AAM0658)
log
On July 5, 2011, at precisely 3:06 AM, an account under the username AAM0658 was observed executing a series of dubious activities that culminated in a significant IT sabotage incident. Initially, the user downloaded a sophisticated keylogger, taking deliberate care to ensure its stealthiness. ....
generated contextual data
predicted class

Figure 3.6: Block diagram testing phase of SecureBERT

The above figure illustrates testing phase for secureBERT. Here, an example is shown, the input is a log file of a user AAM0658 that contains the activities performed by the particular user. It contains all the activities including the logon, email, http, devices and file accesses. This log file is passed into the block of preprocessor which preprocesses the data by cleaning the data, dropping unrequired log events and attributes. The preprocessor separates actions on per day basis, that yields another csv file which contains activities of one day of that particular user. Now the preprocessed file is passed into the finetuned llama3.1 model for the contectual data generation. The output of this phase is context data which is the natural language text that gives clear vision of all the activities performed. The resulted contextual data is input to the trained secureBERT model that predicts the output as threat since this user logged on during work off hour and also showed behaviors like downlodding keylogger, transfering priviledged files. Further the same log was parsed into another classification head for classification between scenario1, scnerio2 and scenario3. The model finally predicted scenario3 as final output.

3.3.3 Data Preprocessing

In this block, the data cleaning and processing are done. The raw log events in CERT dataset contains files like http.csv, email.csv, file.csv, device.csv and logon.csv containing the activities regarding logon/logoff activities, file accesses activities, email related actions of users and visiting urls or uploading, downloading to/from the web. All the log events were combined and attribute ”id” was dropped since it does not add any value for the training data. The combined log events were separated according to the user's actions on a day. Every day log of behaviors of a user are stored in a separate csv file. For example: a log file contains the actions of a user JTM450 of 3 days that includes email, http, file,device and logon events. Now, the preprocessor separates all the behaviors on day basis, which will yield three different csv files, each containing actions/behaviors of user each day.

3.3.4 Fine-Tuning Llama3.1

The LLAMA 3.1 model.released in July 2024, is an advanced version in the LLAMA series, designed for improved text generation, comprehension, and contextual understanding. It builds upon previous versions by incorporating more sophisticated architecture and training methodologies. As of its release, LLAMA 3.1 represents a significant advancement in the language model field.

Customising the large language model (LLM) to produce data that appropriately depicts different insider threat scenarios is a necessary step in fine-tuning the LLaMA 3.1 model for insider threat contextual data creation. In cybersecurity, insider threats—like data exfiltration, privilege misuse, or unusual user behavior—are crucial areas where detection and prevention depend heavily on contextual knowledge. Through fine-tuning, the model is exposed to a specific dataset containing various kinds of threats, enabling it to produce text that is subtle and relevant to the situation. This procedure makes the LLaMA 3.1 model a useful tool for creating synthetic data or assisting cybersecurity research by guaranteeing that it can provide high-quality, domain-specific outputs that closely match actual insider threat scenarios.

The primary motivation behind loading the LLaMA 3.1 model in a 4-bit quantised format is the need to maximise computational efficiency. Quantisation greatly reduces the memory footprint and computational effort by reducing the precision of the model's weights from 16- or 32-bit floating points to 4-bit integers. This reduction is particularly crucial when working with large models, such as LLaMA 3.1, whose fine-tuning can need a lot of resources. 4-bit quantisation is intended to preserve the majority of the model's performance even with the precision loss, enabling quicker and more effective training or inference on hardware with constrained resources. This increases the process's accessibility and usefulness, particularly in settings where memory or processing power are limited.

Model Preparation: Loading the 4-Bit Quantized Model

4-bit quantization is a technique used to reduce the memory and computational requirements of the model.To optimize computational efficiency and reduce memory usage, the LLAMA 3.1 model was loaded in its 4-bit quantized form. This quantization process involves converting the model's weights from standard floating-point precision (usually 16-bit or 32-bit) to 4-bit integers. The quantized model was sourced from the Unsloth repository, which specializes in providing efficient model variants for various applications.The quantization can be expressed as:

$$w_{quant} = \text{round}\left(\frac{w_{orig}}{\Delta}\right) \tag{3.13}$$

where:

- w_{quant} is the quantized weight.
- w_{orig} is the original floating-point weight.
- Δ is the quantization step size.

The quantized weights are then scaled and offset to fit into the 4-bit representation, significantly reducing memory requirements and speeding up computations.

Efficiency Optimization: Flash Attention with Xformers

Flash Attention improves the efficiency of the attention mechanism in transformers by optimizing the computation of attention scores.FlashAttention combined with xFormers plays a crucial role in enhancing the model's ability to process long sequences of text, which are often necessary for understanding complex insider threat scenarios. FlashAttention is an optimized attention mechanism that accelerates the training process by reducing the memory usage and computational overhead associated with the attention layers in transformer models. This is particularly useful when fine-tuning models on large datasets or when dealing with lengthy sequences, which are common in cybersecurity logs or user activity records. xFormers, on the other hand, provides efficient implementations of various sparse and low-rank attention mechanisms, further optimizing the attention computation. Together, FlashAttention and xFormers allow the LLaMA 3.1 model to handle long sequences efficiently, ensuring that it can generate detailed and accurate contextual data relevant to insider threats, without compromising on speed or performance.

The standard attention mechanism is computed as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.14}$$

where:

- Q is the query matrix.
- K is the key matrix.
- V is the value matrix.
- d_k is the dimensionality of the keys.

Flash Attention reduces the complexity of this operation by approximating the attention scores more efficiently:

$$\text{FlashAttention}(Q,K,V) \approx \text{ApproxSoftmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.15}$$

where:

- ApproxSoftmax denotes the approximation method used in Flash Attention to speed up the softmax computation.

Parameter efficient Tuning with LoRA

LoRA (Low-Rank Adaptation) is employed during fine-tuning to further enhance efficiency and prevent overfitting. LoRA introduces low-rank matrices into specific layers of the model, allowing for targeted updates that adapt the model to the specific task of insider threat detection. This method reduces the number of parameters that need to be fine-tuned, enabling the model to learn effectively from smaller datasets without requiring full model retraining. This not only speeds up the fine-tuning process but also minimizes the risk of overfitting, where the model becomes too specialized on the fine-tuning data and loses generalization ability. LoRA's approach is particularly beneficial for large models like LLaMA 3.1, where updating the entire model would be computationally expensive and could degrade performance on other tasks.

LoRA fine-tunes the model efficiently by introducing low-rank matrices into the transformer architecture. The standard weight update in a neural network is given by:

$$W_{new} = W_{orig} - \eta \nabla L \tag{3.16}$$

where:

- W_{new} is the updated weight.
- W_{orig} is the original weight.
- η is the learning rate.
- ∇L is the gradient of the loss function.

In LoRA, the update is applied to low-rank matrices A and B as follows:

$$W_{new} = W_{orig} + A \cdot B^T \tag{3.17}$$

where:

- *A* and *B* are low-rank matrices learned during fine-tuning.

This approach updates a smaller number of parameters while preserving the pre-trained knowledge of the model.

Dataset Preparation for Fine-Tuning: For the dataset to be used in fine tuning, the synthesized data was generated from openai by using ”gpt-4-mini” model. For the data generation purpose, an example of log was given as input to the gpt-4-mini model along with the output and instruction. Similar kind of data was generated by using different instructions and different inputs.

Table 3.4: Sample synthetic data for finetuning llama3.1

	input	**instruction**	**output**	**label**
0	logon,{T7V6-Y1QU17PT-4786GCFW},10/20/2010 20:12...	Convert the given logs to natural language and...	On October 20, 2010, at 20:12, a user with the username YIQU17PT commenced...	scenario1
1	logon,{T8MN-BB900-4786MOEFW},10/20/2010 20:12...	Generate the summary of the given logs.	This employee utilized removable drives,checked emails...	normal
2	logon,{Q9T4-MSOT9-PSWR},10/20/2010 20:12...	Extract and summarize the key information for ...	Over the course of two hours, ending at 18:20, their login records showed...	scenario2
3	logon,{T7V6-Y1QU17PT-4786GCFW},10/20/2010 20:12...	For the given logs, synthesize a similar log a...	This marked a significant deviation from their previous behavior...	scenario1
4	logon,{B3D5-C0JD16NA-6963WQHG},10/06/2010 22:28...	Convert the given logs to natural language and...	The user proceeded to download the keyloggers,transferred it to the supervisors..	scenario3

The table above shows sample of the synthesized data generated by gpt-4-mini model for fine tuning task of llama3.1.

Training for fine-tuning and Inferencing

The model was fine-tuned using 240 samples with batch size per device = 2. Total number of trainable parameters are 41,943,040. The model was trained on 2 epochs, which is shown below

```
trainer_stats = trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 240 | Num Epochs = 2
O^O/ \_/ \    Batch size per device = 2 | Gradient Accumulation steps = 4
\        /    Total batch size = 8 | Total steps = 60
 "-____-"     Number of trainable parameters = 41,943,040
```

The process of fine-tuning is shown in the diagram below:

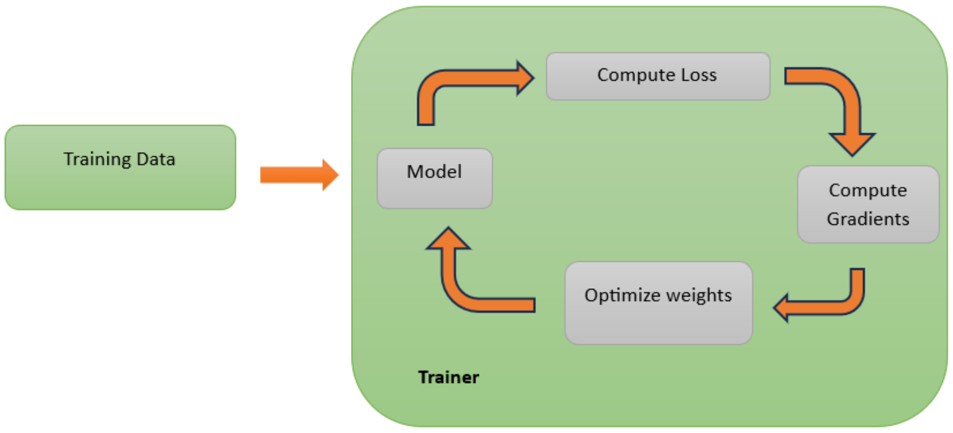
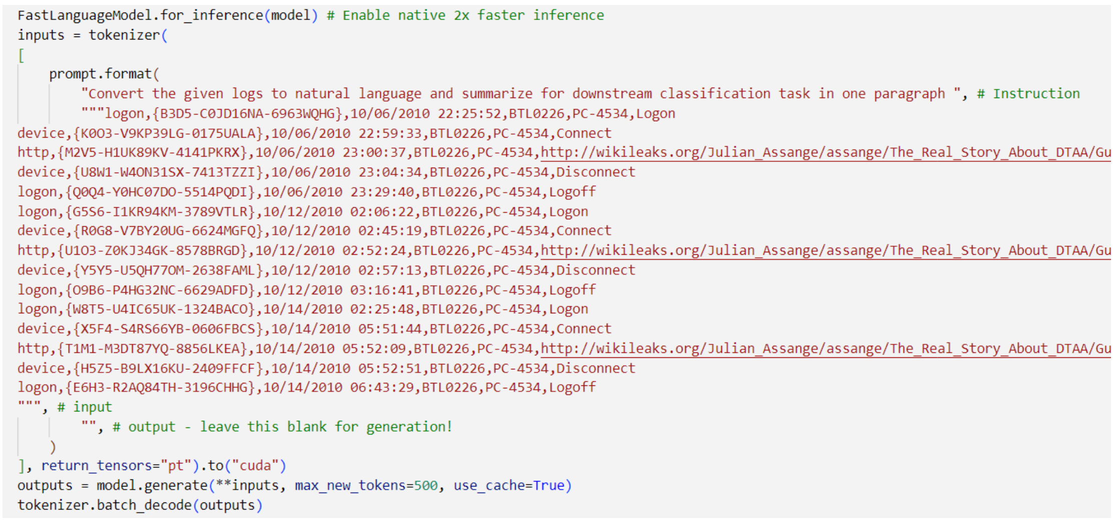


Figure 3.7: Process of fine tuning Llama3.1

After fine tuning the Llama3.1 model, this customed fine-tuned model was used to generate the contextual data that will be input to the secureBERT.

A sample of contextual data generation from fine tuned model is shown in the code snippet below:



```
FastLanguageModel.for_inference(model) # Enable native 2x faster inference
inputs = tokenizer(
[
    prompt.format(
        "Convert the given logs to natural language and summarize for downstream classification task in one paragraph ", # Instruction
        """logon,{B3D5-C0JD16NA-6963WQHG},10/06/2010 22:25:52,BTL0226,PC-4534,Logon
device,{K0O3-V9KP39LG-0175UALA},10/06/2010 22:59:33,BTL0226,PC-4534,Connect
http,{M2V5-H1UK89KV-4141PKRX},10/06/2010 23:00:37,BTL0226,PC-4534,http://wikileaks.org/Julian_Assange/assange/The_Real_Story_About_DTAA/Gu
device,{U8W1-W4ON31SX-7413TZZI},10/06/2010 23:04:34,BTL0226,PC-4534,Disconnect
logon,{Q0Q4-Y0HC07DO-5514PQDI},10/06/2010 23:29:40,BTL0226,PC-4534,Logoff
logon,{G5S6-I1KR94KM-3789VTLR},10/12/2010 02:06:22,BTL0226,PC-4534,Logon
device,{R0G8-V7BY20UG-6624MGFQ},10/12/2010 02:45:19,BTL0226,PC-4534,Connect
http,{U1O3-Z0KJ34GK-8578BRGD},10/12/2010 02:52:24,BTL0226,PC-4534,http://wikileaks.org/Julian_Assange/assange/The_Real_Story_About_DTAA/Gu
device,{Y5Y5-U5QH77OM-2638FAML},10/12/2010 02:57:13,BTL0226,PC-4534,Disconnect
logon,{O9B6-P4HG32NC-6629ADFD},10/12/2010 03:16:41,BTL0226,PC-4534,Logoff
logon,{W8T5-U4IC65UK-1324BACO},10/14/2010 02:25:48,BTL0226,PC-4534,Logon
device,{X5F4-S4RS66YB-0606FBCS},10/14/2010 05:51:44,BTL0226,PC-4534,Connect
http,{T1M1-M3DT87YQ-8856LKEA},10/14/2010 05:52:09,BTL0226,PC-4534,http://wikileaks.org/Julian_Assange/assange/The_Real_Story_About_DTAA/Gu
device,{H5Z5-B9LX16KU-2409FFCF},10/14/2010 05:52:51,BTL0226,PC-4534,Disconnect
logon,{E6H3-R2AQ84TH-3196CHHG},10/14/2010 06:43:29,BTL0226,PC-4534,Logoff
""", # input
        "", # output - leave this blank for generation!
    )
], return_tensors="pt").to("cuda")
outputs = model.generate(**inputs, max_new_tokens=500, use_cache=True)
tokenizer.batch_decode(outputs)
```

The response generated by this fine tuned model can be shown here:

On October 6, 2010, the user logged on to PC-4534 at 10:25 PM and connected to a device at 11:59 PM. They accessed a URL associated with Julian Assange's WikiLeaks at 11:00 PM and the log contained keywords related to espionage, including ”middle-east,” ”isreal,” ”covert,” and ”blackmail.” The user disconnected from the device at 11:04 PM and logged off at 11:29 PM.The user logged on again at 2:06 AM On October 12 and connected to a device at 2:45 AM. They accessed the same URL at 2:52 AM and the log contained similar keywords related to espionage and deceit, including ”lie,” ”covert,” and ”forgery.” The user disconnected from the device at 2:57 AM and logged off at 3:16 AM.

Now, this fine tuned model are used for the dataset generation in contextual format from the raw events in order to train the classification model that is, secureBERT model.

3.3.5 Architecture of Customed Trained SecureBERT Model

The trained secureBERT model lies on the base model which consists of 12 stacked encoder layers, each with embedding size 768. It is trained for the classification task of insider threat detection. It follows the hierarchical training, with classification head added on top of the base model that classifies normal and threat data. If threat is predicted then, it is further trained to classify the threat scenarios. It consists of a classification head. It is trained on the textual generated data from fine tuned llama3.1 model. The generated data consists of 16,800 samples. Among which, 7200 are normal samples and 9600 are malicious/threat samples,3000 of scenario1, 3400 of scenario2 and 3200 of scenario3. The sample data is shown below:

Table 3.5: Sample data for training secureBERT model

	text	**label**
0	On July 25, 2024, at 16:20, a user with the username NZYI7801...	scenario1
1	On July 28, 2024, at 11:13 AM, username KYKP45...	normal
2	On July 5, 2024, at precisely 3:06 AM, an account...	scenario3
3	On July 20, 2024, BVLD4612 logged into their work...	normal
4	On July 23, 2012, at 20:13, user WITB9761 exhibited...	scenario2
...	...	...
	On July 11, 2024, at precisely 05:19, an alarm...	scenario3
16796	On July 24, 2012, an employee using the username...	scenario2
16797	On July 27, 2024, from 05:52 to 13:52, an individual...	scenario3
16798	On July 30, 2024, at precisely 16:39, an incident...	scenario3
16799	Username MINP1179 logged into the system at 01...	normal

The dimensions of head are given below:

Classification head: Consists of:

- hidden layer with dimensions 768 X 768.
- output layer with dimensions 768 X 4.
- probability vector with dimensions 1 X 4.

The custom trained secureBERT architecture is shown in the figure below:

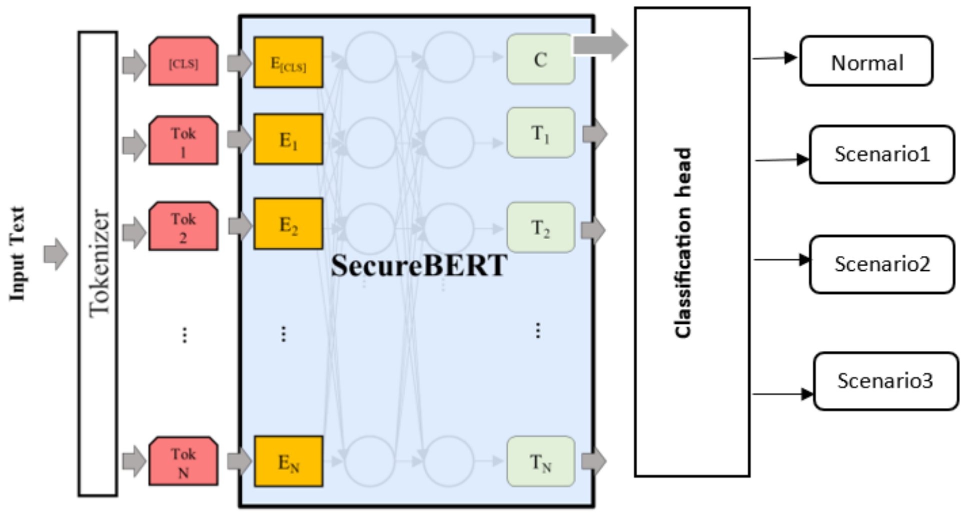


Figure 3.8: Architecture of custom trained secureBERT model

While training the model, L2 regularization is used in the classification head of the secureBERT model, AdamW optimizer is used for efficient and stable training and GridSearchCV is used to tune the hyperparameters of the classification head to achieve the optimal performance. These are explained as:

L2 Regularization

L2 regularization, also known as weight decay, is a technique used to prevent overfitting in machine learning models by penalizing large weights. It adds a regularization term to the loss function that is proportional to the sum of the squared weights.

The modified loss function with L2 regularization can be expressed as:

$$\text{Loss} = \text{Original Loss} + \frac{\lambda}{2}\sum_i w_i^2 \tag{3.18}$$

where:

- λ is the regularization parameter.
- w_i represents the weights of the model.

L2 regularization discourages the model from fitting the training data too closely by shrinking the weights, which helps in reducing the model complexity and improving generalization.

AdamW Optimizer

AdamW is an optimization algorithm that combines the benefits of the Adam optimizer with weight decay regularization. It addresses the shortcomings of the traditional Adam optimizer by decoupling weight decay from the gradient update.

AdamW updates the weights as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{3.19}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{3.20}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{3.21}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.22}$$

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_t \right) \tag{3.23}$$

where:

- g_t is the gradient at time step t.
- m_t and v_t are the first and second moment estimates.
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected moment estimates.
- η is the learning rate.
- λ is the weight decay coefficient.

AdamW ensures that the weight decay is applied correctly by separating it from the gradient-based update, leading to better convergence and performance.

GridSearchCV

GridSearchCV is a scikit-learn hyperparameter optimisation strategy that finds the optimal hyperparameters for a model by conducting an exhaustive search over a given parameter grid.

1. **Parameter Grid:** Define a grid of hyperparameters to search over.
2. **Cross-Validation:** To assess the model's performance, run cross-validation for every combination of hyperparameters.
3. **Selection:** Choose the hyperparameters that yield the best cross-validation performance.

3.4 Instrumentation Requirements

The instrumentation requirements for a User Behavior Analytics (UBA) system, specifically for detecting insider threats using a transformer-based approach, encompass various tools, software, and hardware necessary for efficient development, testing, and deployment. Below is a comprehensive overview of the instrumentation requirements:

3.4.1 Development Environment

- **Integrated Development Environment (IDE):**
  - Jupyter Notebook: An interactive IDE widely used for writing and testing code, particularly in data science and machine learning projects. It allows for inline visualization and step-by-step execution, which is crucial for debugging complex models.
  - PyCharm: Another popular IDE, especially for Python development, which offers advanced debugging capabilities, integrated testing, and powerful code navigation.
- **Version Control System:**
  - Git: A distributed version control system essential for tracking changes in the codebase, collaborating with other developers, and managing multiple branches of development.
  - GitHub/GitLab: Platforms for hosting Git repositories, enabling seamless collaboration and continuous integration/continuous deployment (CI/CD) pipelines.
- **Text Editor:**
  - Visual Studio Code: A lightweight, yet powerful text editor for manually editing and inspecting code files, with extensive plugin support for various languages and frameworks.

3.4.2 Programming Languages and Libraries

- **Python:**
  - The primary programming language used for developing machine learning models, data preprocessing pipelines, and overall system implementation due to its rich ecosystem of libraries and community support.

- **Machine Learning Frameworks:**
  - TensorFlow: A robust framework developed by Google for building and training deep learning models, particularly useful for large-scale neural networks.
  - PyTorch: An open-source machine learning library developed by Facebook's AI Research lab, known for its dynamic computation graph, making it easier to debug and iterate on models.
  - Keras: A high-level neural networks API, running on top of TensorFlow, that simplifies the process of designing, training, and deploying deep learning models.
- **Natural Language Processing (NLP) Libraries:**
  - NLTK (Natural Language Toolkit): A suite of libraries and programs for symbolic and statistical natural language processing (NLP) in Python, widely used for tasks like tokenization, tagging, parsing, and classification.
  - Hugging Face Transformers: A popular library for leveraging pre-trained transformer models, such as BERT, GPT, and others, for various NLP tasks including text classification, summarization, and translation.
- **Data Manipulation and Analysis Libraries:**
  - Pandas: A powerful data manipulation and analysis library in Python, providing data structures like DataFrames to handle large datasets efficiently.
  - NumPy: A fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them.
  - scikit-learn: A machine learning library that provides simple and efficient tools for data mining and data analysis, built on NumPy, SciPy, and matplotlib.

    - Matplotlib: A comprehensive library for creating static, animated, and interactive visualizations in Python, commonly used for generating plots and charts for data analysis.

3.4.3 Hardware Requirements

- **High-Performance Computing Resources:**
    - **GPUs (Graphics Processing Units):**
        * L4 GPU: A mid-range graphics card suitable for accelerating deep learning computations during the development and testing phases of the project.
        * A100 GPU: A high-end, data center-class GPU designed by NVIDIA, optimized for training large-scale neural networks and handling the immense computational demands of transformer-based models.
    - **RAM and Storage:**
        * Sufficient RAM (typically 32GB or more) to load large datasets and run memory-intensive operations.
        * High-capacity storage solutions (such as SSDs) to store vast amounts of training data, model checkpoints, and other necessary files.
    - **Cloud Computing Resources:**
        * Amazon Web Services (AWS): Provides scalable computing power and storage solutions, enabling the use of high-performance GPUs and CPUs on demand. Services like EC2 and S3 are particularly useful for managing large-scale machine learning workloads.
        * Google Colab Pro: A cloud-based platform that offers enhanced computing resources, including access to premium GPUs (such as Tesla T4, P100, and even A100 for Pro+ users), which is particularly valuable for training and fine-tuning deep learning models.

3.4.4 APIs and External Services

- **OpenAI's GPT-4 Mini:**
  - Used for synthesizing data required for fine-tuning the LLAMA 3.1 model. Accessing this model involved using OpenAI's API and endpoint on a pay-per-use basis, allowing for dynamic data generation tailored to the specific needs of the insider threat detection system.

For the fine-tuning of the LLAMA 3.1 model, substantial computational resources were necessary, including access to premium GPUs such as the L4 and A100. These were accessed via Google Colab Pro subscription, which provides a pay-per-use model, allowing for the flexibility to scale up resources as needed during the intensive training phases.

3.5 Dataset Explanation

The CERT (Computer Emergency Response Team) dataset was introduced by the CERT Division of the Software Engineering Institute (SEI) at Carnegie Mellon University. The CERT Division is known for its work in cybersecurity, including research on insider threats. The dataset is part of their efforts to provide realistic data for the research community to study insider threat scenarios and develop effective detection and prevention methods. The CERT Insider Threat Dataset, specifically versions like 4.2, is widely used for benchmarking and developing models for insider threat detection.

The primary advantage of synthetic data over real datasets, which must employ masking techniques that degrade information quality, is that it has no privacy limitations at all. The CERT dataset also improves collaboration by offering a research standard that anyone can use to compare their insider threat detection models, which is one of the reasons of choosing it in this project. Additionally, there are files containing personal data from the users. The LDAP files (Lightweight Directory Access Protocol) used to extract roles within the company, psychometric values, which are not currently used in this project. Selecting which of the several CERT data releases to employ is a crucial task because comparable research has not produced a consensus. The CERT dataset is released in many versions with different numbers of attacks, users, lengths, and advanced text generation techniques, among other, version 4,2 was chosen, which had 1000 users and 501 days of activity data, 70 of whom carried out malicious behaviors. The r4.2 encompasses more than 32 million potential possibilities, comprising 7327 malicious actions.

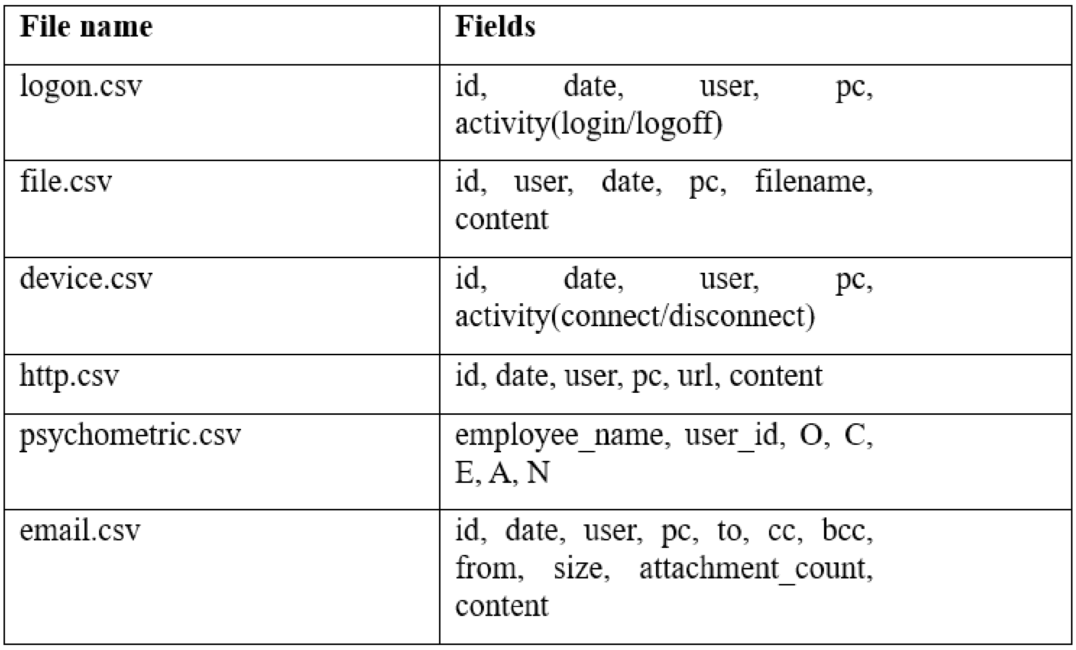
The five log files that make up the CERT 4.2 dataset each contain raw events that list a specific type of activity for every user. The collection contains five different kinds of raw events or activities. They are listed as follows:

- logon.csv: contains the records of logon or logoff activities by each employee
- file.csv: has information by the file access made by the employees
- device.csv: contains events regarding the devices accessed by users
- email.csv: has the information of events about email transactions between employee
- http.csv: contains the information of url visited by each employee

The CERT 4.2 dataset is chosen for this project, due to several compelling reasons:

- **Comprehensive and Realistic Data**: The CERT 4.2 dataset is one of the most comprehensive insider threat datasets available. It includes a wide variety of user activities such as logons, device usage, file access, email communication, and web browsing. This diversity allows for a more realistic and holistic analysis of insider threats.
- **Scenario-Based Data**: The dataset includes multiple predefined scenarios that simulate realistic insider threat activities. These scenarios are crafted to reflect actual malicious behaviors and provide a robust framework for testing and evaluating threat detection models.
- **Well-Documented**: The CERT datasets are well-documented, with clear descriptions of the scenarios and activities. This transparency facilitates easier understanding and usage of the data for research purposes.

Table 3.6: Attribute fields of each log file



File name	Fields
logon.csv	id, date, user, pc, activity(login/logoff)
file.csv	id, user, date, pc, filename, content
device.csv	id, date, user, pc, activity(connect/disconnect)
http.csv	id, date, user, pc, url, content
psychometric.csv	employee_name, user_id, O, C, E, A, N
email.csv	id, date, user, pc, to, cc, bcc, from, size, attachment_count, content

The CERT 4.2 dataset includes several insider threat scenarios. Here is a detailed description of the key scenarios:

- **Scenario 1:** Represents the behaviour of the user who was not working after the work duration, did not use the removable devices, starts working after the work time, uses removable media, uploads the confidential data into wikileaks.org, and then leaves the company.

- **Scenario 2:** Represents the user's behaviour by displaying his activity of looking through recruitment sites and approaching a rival company about a job offer. Before quitting the company, they stole data using a thumb drive (far more frequently than before).

- **Scenario 3:**Shows the following user actions- The system administrator gets irate. downloads a keylogger and transfers it to his boss's computer using a thumb drive. He signs in as his supervisor the following day using the keylogs he has gathered, sending out a frightening mass email that sends shivers down the spine of the company. He quits the company right away.

Table 3.7: CERT4.2 event counts

Event Type	Number of events
Logging in and out (logon.csv)	854,860
Using pendrives (device.csv)	405,380
Email traffic (email.csv)	2,629,980
Www traffic(http.csv)	28,434,423
File Operations(fie.csv)	445,581

3.5.1 Statistics of Dataset

The number of events in each of the log files are shown in Table3.6. Among all the total events, only events 31 days were extracted for all the users. A directory was created for all of the users. Each directory contains the activities performed by that user in a day in separate csv files. These csv files are then fed into the fine-tuned Llama3.1 model which generated the contextual data from each input per day log events. These generated contextual data are then fed into the secureBERT model for the classification purpose.

3.5.2 Dataset for Llama3.1 Fine-Tuning

The dataset used for fine-tuning the LLAMA 3.1 model comprises 2,400 synthesized samples generated using OpenAI's GPT-4 Mini. This dataset is organized into three primary columns: Input (Logs), Output (Contextual Data), and Label.

Input (Logs) includes raw log data, capturing user activities, system events, and security alerts. These logs typically contain timestamps, user IDs, event types, and other relevant metadata. The logs serve as the factual foundation for generating meaningful narratives, focusing on user behavior and system interactions.

Output (Contextual Data) is generated by GPT-4 Mini, transforming raw logs into natural language descriptions that highlight key events, patterns, and potential security threats. This contextualization includes summarization, pattern recognition, and the provision of context, making it easier for security analysts to interpret the data.

Label categorizes the contextual data into normal or scenario-based labels, such as ”Scenario1,” ”Scenario2,” etc., which correspond to different insider threat scenarios.

This categorization helps the model learn to distinguish between typical and potentially malicious behavior. This fine-tuning process enhances the LLAMA 3.1 model's ability to generate insightful narratives and accurately classify activities, enabling more effective insider threat detection and response.

3.5.3 Dataset for Training secureBERT model

The dataset generated by the fine-tuned LLAMA 3.1 model comprises 16,800 samples that are categorized into normal and malicious activities. Among these, 7,200 samples are labeled as ”Normal,” representing benign user activities that do not indicate any security threats. The remaining 9,600 samples are classified into three distinct threat scenarios: Scenario 1, Scenario 2, and Scenario 3. This distribution includes 3,000 samples for Scenario 1, 3,400 for Scenario 2, and 3,200 for Scenario 3. Each sample includes textual descriptions of user activities, reflecting various contexts in which potential insider threats could manifest.

This dataset is crucial for training and evaluating security models, particularly those focused on detecting insider threats. The normal samples provide a baseline of expected user behavior, while the scenario-based malicious samples simulate different types of insider threats, helping models learn to distinguish between typical and anomalous activities. The detailed labeling and diverse scenarios enable comprehensive training, allowing models to better identify and respond to various threat patterns in real-world applications.

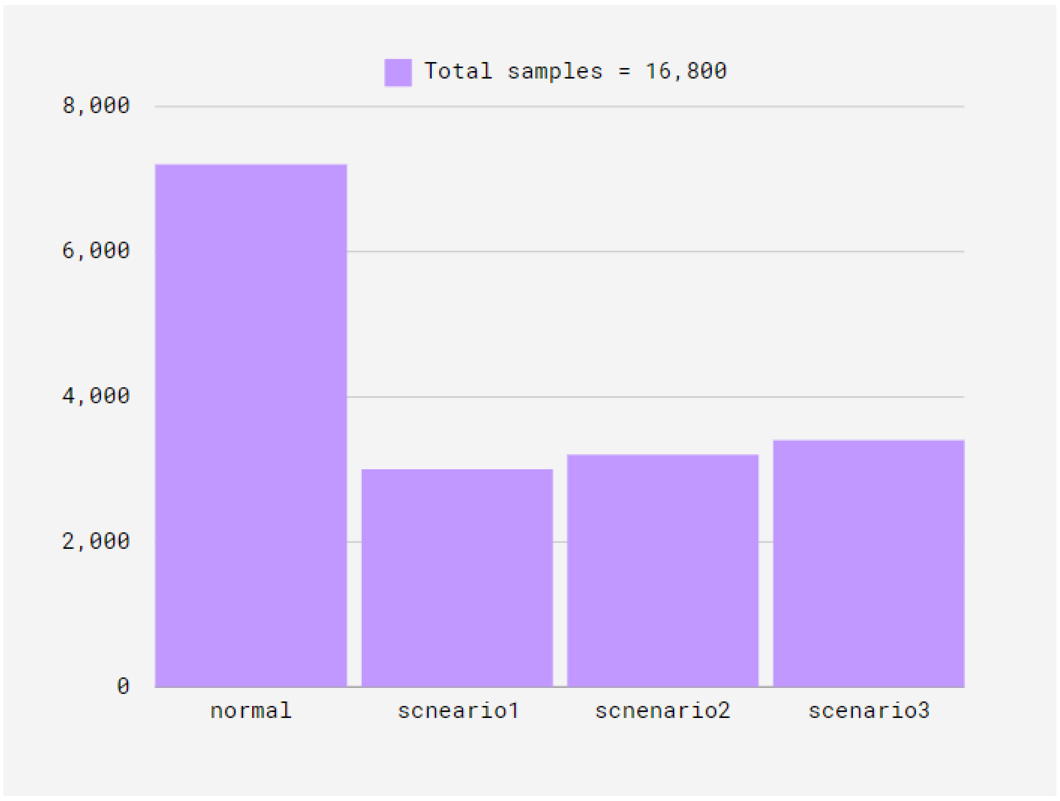


Figure 3.9: Histogram representing the dataset for secureBERT training

3.6 Description of Algorithms

3.6.1 Algorithm Description for Fine-Tuning LLAMA 3.1 Model

Algorithm 1 Fine-Tune LLAMA 3.1 Model

```
Require: $pretrained\_model$: Pretrained LLAMA 3.1 model, $quantized\_weights$: 4-bit quantized weights, $flash\_attention$: Flash Attention with xformers, $lora\_parameters$: LoRA parameters, $training\_data$: Training data samples
Ensure: $fine\_tuned\_model$: Fine-tuned LLAMA 3.1 model
 1: $model \leftarrow$ initialize_model($pretrained\_model$)    ▷ Initialize LLAMA 3.1 model
 2: $model \leftarrow$ load_weights($model, quantized\_weights$)    ▷ Load 4-bit quantized weights
 3: $flash\_attention\_module \leftarrow$ initialize_flash_attention()    ▷ Initialize Flash Attention module
 4: $model \leftarrow$ integrate_flash_attention($model, flash\_attention\_module$)    ▷ Integrate Flash Attention with LLAMA 3.1 model
 5: $model \leftarrow$ apply_lora($model, lora\_parameters$)    ▷ Apply LoRA for parameter-efficient fine-tuning
 6: $training\_data \leftarrow$ load_training_data($training\_data$)    ▷ Load and preprocess training data
 7: $training\_loop \leftarrow$ initialize_training_loop($model, training\_data$)    ▷ Set up training loop
 8:
 9: while $training\_loop.has\_more\_epochs()$ do    ▷ Loop through epochs
10:     $batch \leftarrow training\_loop.get\_next\_batch()$    ▷ Get next batch of data
11:     $outputs \leftarrow$ model.forward($batch.inputs$)    ▷ Perform forward pass
12:     $loss \leftarrow$ compute_loss($outputs, batch.targets$)    ▷ Compute loss
13:     model.backward($loss$)    ▷ Perform backward pass
14:     update_model_parameters()    ▷ Update model parameters using optimizer
15: end while
16: $fine\_tuned\_model \leftarrow$ save_model($model$)    ▷ Save the fine-tuned model
17: return $fine\_tuned\_model$
```

The fine-tuning process for the LLAMA 3.1 model begins by initializing the model with pretrained weights and applying 4-bit quantized weights for efficiency. The process incorporates Flash Attention with xformers to enhance model performance and applies LoRA (Low-Rank Adaptation) parameters to enable efficient fine-tuning. Training data is loaded and preprocessed, setting up a training loop that iterates through multiple epochs. During each epoch, the model performs forward passes with the data, computes and backpropagates the loss, and updates model parameters using an optimizer. Finally, the fine-tuned model is saved for subsequent use, completing the training process.

3.6.2 Algorithm Description for Generating Contextual Data from Fine-Tuned LLAMA 3.1 Model

Algorithm 2 Generate Contextual Data from Fine-Tuned LLAMA 3.1 Model

```
Require: fine_tuned_model: Fine-tuned LLAMA 3.1 model, log_data: Input log data
Ensure: contextual_sentences: Generated contextual data sentences
1:  processed_logs ← preprocess_logs(log_data)    ▷ Preprocess log data for model input
2:  contextual_sentences ← []    ▷ Initialize empty list for storing sentences
3:
4:  for each log entry in processed_logs do    ▷ Iterate through each log entry
5:      input_text ← convert_log_to_text(log_entry)    ▷ Convert log entry to text format
6:      generated_text ← fine_tuned_model.generate(input_text)    ▷ Generate contextual text using the model
7:      contextual_sentences.append(generated_text)    ▷ Append generated text to list
8:  end for
9:  contextual_sentences ← postprocess_sentences(contextual_sentences)    ▷ Postprocess generated sentences for final output
10: return contextual_sentences
```

This algorithm generates contextual data using a fine-tuned LLAMA 3.1 model by first preprocessing the input log data to make it suitable for model input. It initializes an empty list to store the generated sentences. For each log entry, it converts the log into a text format, uses the fine-tuned LLAMA 3.1 model to generate contextual text, and appends this text to the list. After processing all log entries, it performs post-processing on the generated sentences to prepare them for final output, and then returns the list of contextual sentences.

3.6.3 Algorithm for Training the SecureBERT model

The algorithm for training the secureBERT model for multiclass classification involves several key steps. It begins by initializing the model with pretrained weights, setting up an optimizer with a specified learning rate, and selecting a loss function appropriate for multiclass classification, such as cross-entropy loss.

The training process is structured over multiple epochs. In each epoch, the model processes batches of training data. For each batch, the model performs a forward pass to generate predictions, calculates the loss based on the predictions and actual targets, and then performs backpropagation to compute gradients. These gradients are used to update the model parameters through the optimizer.

After completing the training for each epoch, the model is evaluated on a separate validation dataset to assess its performance. If the model's accuracy on the validation set improves, it updates the record of the best accuracy and saves the current best model.

The training continues through all specified epochs, and at the end of the process, the model with the best performance on the validation set is returned. This ensures that the final model is the one that achieved the highest accuracy during training and validation.

Algorithm 3 Train SecureBERT Model for Multiclass Classification

```
Require: $pretrained\_securebert$: Pretrained secureBERT model, $training\_data$: Training data, $validation\_data$: Validation data, $learning\_rate$: Learning rate, $batch\_size$: Batch size, $num\_epochs$: Number of epochs
Ensure: $best\_model$: Trained secureBERT model with highest validation accuracy
 1: $model \leftarrow$ initialize_model($pretrained\_securebert$)            ▷ Initialize model
 2: $optimizer \leftarrow$ initialize_optimizer($model, learning\_rate$)      ▷ Set up optimizer
 3: $loss\_function \leftarrow$ initialize_loss_function(cross_entropy)      ▷ Set loss function
 4: $best\_accuracy \leftarrow 0$                                          ▷ Initialize best accuracy
 5: for epoch from 1 to $num\_epochs$ do
 6:     for each batch in $training\_data$ do
 7:         $inputs, targets \leftarrow$ batch                                ▷ Get batch data
 8:         $outputs \leftarrow$ model.forward($inputs$)                      ▷ Forward pass
 9:         $loss \leftarrow$ loss_function($outputs, targets$)               ▷ Compute loss
10:         optimizer.zero_grad()                                          ▷ Reset gradients
11:         loss.backward()                                                ▷ Backpropagate gradients
12:         optimizer.step()                                               ▷ Update parameters
13:     end for
14:     $accuracy \leftarrow$ evaluate_model($model, validation\_data$)       ▷ Evaluate model
15:     if $accuracy > best\_accuracy$ then
16:         $best\_accuracy \leftarrow accuracy$                              ▷ Update best accuracy
17:         $best\_model \leftarrow$ save_model($model$)                      ▷ Save best model
18:     end if
19: end for
20: return $best\_model$
```

3.7 Verification and Validation Procedures

A confusion matrix is a tool used to evaluate the performance of a classification algorithm by displaying the actual versus predicted classifications. It helps visualize how well the classifier is performing, especially in identifying different types of errors.

Structure of the Confusion Matrix

A confusion matrix is typically a square matrix where the number of rows and columns corresponds to the number of classes in the classification problem. For a binary classification problem, it looks like this:

Table 3.8: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

True Positive (TP): The count of cases accurately identified as positive.

True Negative (TN): The count of cases accurately identified as negative.

False Positive (FP): The count of cases wrongly identified as positive (also referred to as Type I error).

False Negative (FN): The count of cases wrongly identified as negative (also referred to as Type II error).

1. **Accuracy:** The proportion of correctly classified instances (both true positives and true negatives) out of the total instances

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.24}$$

2. **Precision (Positive Predictive Value):** The proportion of positive predictions that are actually correct.

$$Precision = \frac{TP}{TP+FP} \tag{3.25}$$

3. **Recall (Sensitivity or True Positive Rate):** The proportion of actual positives that are correctly identified.

$$Recall = \frac{TP}{TP+FN} \tag{3.26}$$

4. **Specificity (True Negative Rate):** The proportion of actual negatives that are

correctly identified.

$$Specificity = \frac{TP}{TP + FN} \tag{3.27}$$

5. **F1-Score** The harmonic mean of precision and recall, providing a balance between the two.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.28}$$

ROC Curve

A graphical representation called the Receiver Operating Characteristic (ROC) curve is used to assess how well a binary classification model performs. It shows the trade-off between the model's propensity to classify negative examples as positive (False Positives) across a range of decision thresholds and its ability to accurately identify positive instances (True Positives).

The ROC curve is plotted with the FPR on the x-axis and the TPR on the y-axis. Each point on the ROC curve corresponds to a different threshold used to classify the positives and negatives.The area under the ROC curve (AUC) is a single scalar value that summarizes the overall performance of the model. An AUC of 1.0 indicates a perfect classifier, while an AUC of 0.5 indicates a model with no discriminatory power. It is typically calculated using numerical integration methods because the ROC curve is a plot of discrete points (FPR, TPR) for different thresholds. The most common method to calculate the AUC is the Trapezoidal Rule, which approximates the area under the curve as a series of trapezoids and sums their areas.

The Area Under the Curve (AUC) for a ROC curve can be calculated using the Trapezoidal Rule, which is defined as:

$$\text{AUC} = \sum_{i=1}^{n-1} \frac{(\text{TPR}_{i+1} + \text{TPR}_i)}{2} \times (\text{FPR}_{i+1} - \text{FPR}_i) \tag{3.29}$$

where:

- n is the number of points on the ROC curve.
- TPR_i and FPR_i are the True Positive Rate and False Positive Rate at the i^{th}

threshold.

Interpretations of ROC

- Perfect Classifier: A model that perfectly distinguishes between the two classes will have a ROC curve that passes through the top left corner (TPR = 1, FPR = 0) and an AUC of 1.
- Random Classifier: A classifier that makes random guesses will have a ROC curve that lies along the diagonal line (from (0, 0) to (1, 1)), with an AUC of 0.5.
- Better-Than-Random Classifier: A ROC curve that is above the diagonal indicates better-than-random performance, and the closer the curve is to the top left corner, the better the classifier.

4 RESULTS

4.1 Training Loss curve of the Llama3.1 Model

The Llama 3.1 model was fine-tuned for the contextual data generation task from the structured data. This fine tuning of the model was initiated by loading the 8 billion parameter version in 4 bit quantization. The model was configured with Parameter efficient Fine-Tuning using LoRA(Low-Rank Adaptation) to fine-tune the model by updating specific projection layers (q_proj, k_proj, etc.) with a low-rank adaptation (r=16) and scaling (lora_alpha=16), no dropout (lora_dropout=0), no bias tuning, and using gradient checkpointing.The model was fine-tuned with learning rate = 2e-4, using AdamW optimizer and weight decay = 0.01. The training loss obtained during the process can be seen in the figure below.

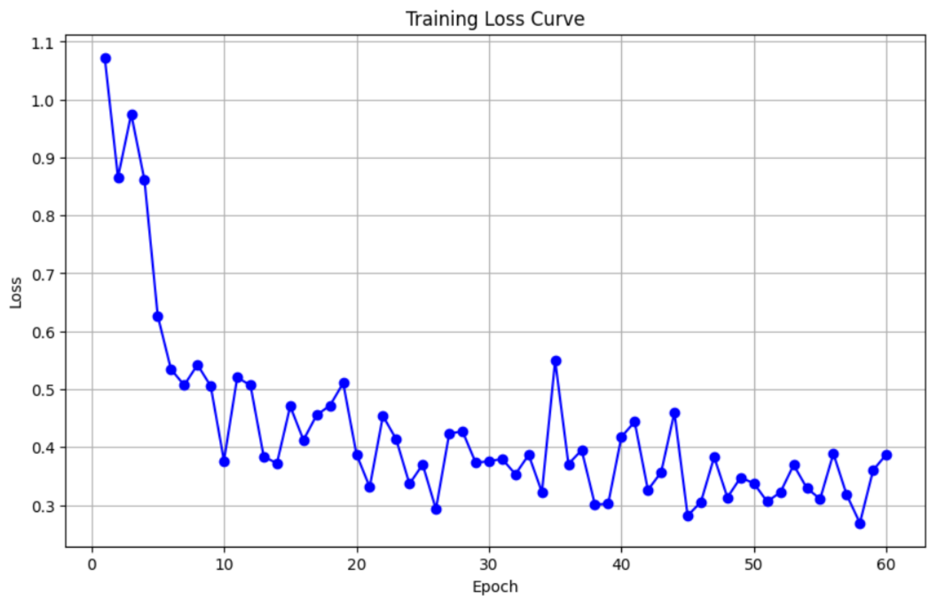


Figure 4.1: The training loss obtained while fine-tuning the Llama3.1 model

The graph shows the training loss obtained when fine tuning the LLaMA 3.1 model over 60 epochs for a contextual data generation. As seen on the graph, the values of loss in epoch 0 are greater than 1, but they rapidly decrease to about 0.4 within the first ten epochs. These results suggest that the model is converging though it has some discrepancies or noises during training. Since we observe a trend towards zero in this period with some fluctuations, we can say that model is converging despite some noise or changes during training. The trend suggests successful fine-tuning.

4.2 Contextual Data Generation from Fine-Fined Llama3.1

Here, is an example of logs representing each scenario and the corresponding contextual data for the particular log, which is generated by the fine-tuned Llama model.

Normal:

```
logon {GBGS-941190LT-1806JFGT} 4/24/2011 06:18 BSS0369 PC-8486
Logon
email {GBGS-9411EM1L-0217EML1} 4/24/2011 06:25 BSS0369 PC-8486
GBGS9411@dtaa.com
device {GBGS-9411USB1-1000USB1} 4/24/2011 10:00
BSS0369 PC-8486 Connect USB drive Drive G:\ 0 0
USB drive connected for file transfer
file {GBGS-9411TRF1-1005TRF1} 4/24/2011 10:05
BSS0369 PC-8486 Transfer Report.docx Drive G:\ 0 0
Transferred report document to USB drive
http {GBGS-9411URLV-1030URLV} 4/24/2011 10:30 BSS0369
PC-8486 http://www.companyportal.com/project-update
Visited company portal for project updates
device {GBGS-9411USB2-1300USB2} 4/24/2011 13:00
BSS0369 PC-8486 Disconnect USB drive Drive G:\ 0 0
USB drive disconnected after file transfer
logon {GBGS-9411LOGF-1318LOGF} 4/24/2011 13:18 BSS0369 PC-8486
Logoff
```

The above log sample shows a sequence of user activity on 24th April,2011. The user performed various activities during his/her session time. As seen, the log is in structured format and will be converted to the corresponding contextual data, which is the output from Llama3.1 model. The contextual data corresponding to the above log is:

On April 24, 2011, user GBGS9411 logged into their computer at 06:18 to kick off a productive day in the office. Almost immediately, the user began with a routine check of their emails, sifting through a total of 17 messages that required attention. Following this, at 08:00, they participated in the first of three scheduled meetings, where team members discussed project deadlines and key deliverables. After the meetings, GBGS9411 dedicated time to creating a detailed report that summarized the outcomes of recent tasks, ensuring that stakeholders were informed of the progress. Throughout the day, collaboration was key; the user engaged in four different discussions with colleagues to brainstorm ideas and seek input on ongoing projects. As lunch approached, they took a moment to reflect on the morning's work, realizing how these typical office activities—much like the patterns observed in their past workdays—coalesced into a fulfilling work routine. By 13:18, they wrapped up for the day, feeling accomplished and ready for the challenges ahead.

Scenario1: As far as we are aware, scenario 1 is the set of events that characterizes a user who started logging in after hours, using a detachable drive, and uploading data to wikileaks.org. This user, who had never used a removable drive or worked after hours, departs the company soon after. Let's see an example log of this scenario along with the corresponding contextual data.

```
email {T1T8-D7NK22SC-0592NDUJ} 9/30/2010 13:31
BSS0369 PC-3672 Francis.Brian.Armstrong@dtaa.com
Brenden.Samuel.Shaffer@dtaa.com Brenden.Samuel.Shaffer@dtaa.com
26741 0 company will suffer i may leave no gratitude
email {U7F0-R9UM23RL-9942VQSR} 9/30/2010 13:34 FBA0348 PC-8486
Brenden.Samuel.Shaffer@dtaa.com Francis.Brian.Armstrong@dtaa.com
27766 0 schedule lets talk hard job rest valued employee lets talk
email {G5Y6-S2IS77AI-5592FENQ} 9/30/2010 14:32
BSS0369 PC-3672 Francis.Brian.Armstrong@dtaa.com
Brenden.Samuel.Shaffer@dtaa.com Brenden.Samuel.Shaffer@dtaa.com
```

```
37158 0 outraged outraged bad things not my fault
email {J7S0-J2BF54BJ-1560ZJFB} 9/30/2010 14:44 FBA0348 PC-8486
Brenden.Samuel.Shaffer@dtaa.com Francis.Brian.Armstrong@dtaa.com
27715 0 operose assistance unrelenting demanding job lets talk today
http {U4O1-O1IE61HB-3238USSY} 9/30/2010 16:02 BSS0369 PC-3672
http://www.relytec.com/Climate_of_Florida/suniest/NNXrlybttre1893061997.aspx
device {P0D1-D3EP98MZ-5216UZLY} 9/30/2010 16:07 BSS0369 PC-3672
Connect
file {A5G9-U0DS24RQ-5724QSCY} 9/30/2010 16:10 BSS0369 PC-3672
TBDZA9ME.exe
device {F0B5-B3GP27FZ-1059KQUY} 9/30/2010 16:15 BSS0369 PC-3672
Disconnect
logon {O4B2-O6CO48UA-3416FAAM} 9/30/2010 20:16 BSS0369 PC-8486
Logon
device {S6U3-G1GM61UU-6821UTNG} 9/30/2010 20:26 BSS0369 PC-8486
Connect
device {L5V0-T7KT12VV-5753XJEO} 9/30/2010 20:30 BSS0369 PC-8486
Disconnect
logon {F3A4-G8RG38TH-4582NBDR} 9/30/2010 20:31 BSS0369 PC-8486
Logoff
logon {F5U1-K4NZ89MG-2416CBXD} 10/1/2010 19:19 BSS0369 PC-8486
Logon
logon {C7V6-T5HQ75BY-8395TCFN} 10/1/2010 19:33 BSS0369 PC-8486
Logoff
logon {F2X9-N6CT67WV-0165DWKQ} 10/1/2010 19:40 FBA0348 PC-8486
Logon
email {A8K0-Q3DX55DM-3791QCFI} 10/1/2010 19:45 FBA0348 PC-8486
Arthur.John.Floyd@dtaa.com;Francis.Brian.Armstrong@dtaa.com 47278 0
reassign funding reallocate
```

```
logon {B1I2-X3TE24CB-4083EMHJ} 10/1/2010 19:48 FBA0348 PC-8486
Logoff
```

The corresponding contextual data for above log, generated by the fine-tuned model is given as:
"On the evening of September 30, 2010, at 13:31, a concerning shift in behavior began to unfold for an employee identified as Brenden Samuel Shaffer. Over the course of the next six hours, he engaged in a series of alarming activities that starkly contrasted with his previous work patterns. Initially, he sent a total of 16 emails to his supervisor, which included 13 messages that were flagged as urgent, indicating a sudden increase in his communication frequency. This change in behavior was particularly notable given that in the past, Brenden had never utilized the urgent flag before. As the night progressed, his actions became even more concerning, as he began downloading a significant number of files, including a suspicious piece of software that raised red flags for potential malicious intent. During this time, he also accessed his supervisor's account without authorization, a clear violation of company policy. The combination of these actions suggested a calculated effort to manipulate information and potentially undermine organizational trust. In a further alarming development, Brenden connected a removable drive to his workstation, which could indicate a possible attempt to exfiltrate sensitive data from the company. This act, coupled with the previous unauthorized access, escalated the severity of the situation. On October 1, 2010, at 19:19, Brenden logged in for the first time, marking the beginning of a new workday that would ultimately lead to his departure from the organization. Throughout this session, he sent 15 emails to various colleagues, which included a few that contained misleading information, further contributing to the chaos. The culmination of these actions took place when Brenden sent an email to the entire organization, which included a mass resignation of funding and intention to down size, effectively causing a significant impact on the team's morale and productivity. As the night wore on, Brenden logged in a total of 14 times, indicating a clear pattern of after-hours activity that deviated sharply from his typical work habits. On the following day, October 2, 2010, he left the organization without notice, leaving behind a trail of

confusion and concern among his colleagues and management. The activities exhibited by Brenden Samuel Shaffer from 13:31 on September 30, 2010, to 19:19 on October 1, 2010, paint a concerning picture of potential insider threat behavior. The combination of urgent emails, unauthorized access, removable drive usage, and the sudden departure from the organization raise significant alarms regarding the security and integrity of the company's data and operational processes."

Scenario2: As we know, scenario2 can be defined as the sequence of activities of an employee, who starts browsing job websites and seeking employment from a competitor, significantly increases their use of a thumb drive to steal data before leaving the company. An example of log representing this scenario is:

```
7/23/2012 20:13|WITB9761|PC 1234|HTTP|http://jobhuntersbible.com
7/23/2012 20:20|WITB9761|PC-1234|HTTP| http://indeed.com
7/23/2012 20:30|WITB9761|PC-1234| HTTP|http://monster.com
7/23/2012 20:45 | WITB9761 | PC-1234 | Device | Activity: Connect
7/23/2012 20:50 | WITB9761 | PC-1234 | Device | Activity: Disconnect
7/23/2012 20:55 |WITB9761|Email|user_WITB9761@dtaa.com 20480
7/23/2012 21:00 | WITB9761 | PC-1234 | Device | Activity: Connect
7/23/2012 22:00 | WITB9761 | PC-1234 | Logon
```

Now, the corresponding contextual data for above log is:

"On July 23, 2012, at 20:13, user WITB9761 exhibited highly suspicious behavior indicative of potential intellectual property theft. Over the course of two hours, the user accessed job search websites 28 times, a markedly higher frequency than typical activity patterns observed for employees within the organization. During this session, records indicate that the user sent 3 emails explicitly addressing a competitor, raising alarms about possible information sharing that could jeopardize proprietary secrets. Furthermore, the user connected USB drives to the company's systems on 3 separate occasions, suggesting an intention to transfer sensitive data outside the controlled network. Notably, the system logs revealed that WITB9761 engaged in large data transfers, effectively circumventing internal safeguards designed to track such actions. Additionally, the user demonstrated a worrying trend of logging in outside of regular hours, with 7 instances noted on that day alone, which could indicate an effort to avoid detection while executing their plans. Compounding the unsettling evidence, the user copied 17 proprietary files, all of which are intrinsically valuable to the company's competitive edge. This combination of job hunting, communication with competitors, external device usage, and significant data transfers paints a concerning picture of WITB9761's intentions, warranting immediate investigation into their activities for potential breaches of trust and security protocols."

Scenario3: The scenario 3 can be defined as "A disgruntled system administrator downloads a keylogger, transfers it to his supervisor's machine using a thumb drive, and, the following day, uses the collected keylogs to impersonate his supervisor, sending a panic-inducing mass email before leaving the organization." Lets look into an example representing this scenario.

```
email {T1T8-D7NK22SC-0592NDUJ} 9/30/2010 13:31
BSS0369 PC-3672 Francis.Brian.Armstrong@dtaa.com
Brenden.Samuel.Shaffer@dtaa.com Brenden.Samuel.Shaffer@dtaa.com
26741 0 company will suffer i may leave
email {U7F0-R9UM23RL-9942VQSR} 9/30/2010 13:34 FBA0348 PC-8486
Brenden.Samuel.Shaffer@dtaa.com Francis.Brian.Armstrong@dtaa.com
27766 0 schedule lets talk hard job rest valued employee
email {G5Y6-S2IS77AI-5592FENQ} 9/30/2010 14:32
BSS0369 PC-3672 Francis.Brian.Armstrong@dtaa.com
Brenden.Samuel.Shaffer@dtaa.com Brenden.Samuel.Shaffer@dtaa.com
37158 0 outraged outraged bad things not my fault
email {J7S0-J2BF54BJ-1560ZJFB} 9/30/2010 14:44 FBA0348 PC-8486
Brenden.Samuel.Shaffer@dtaa.com Francis.Brian.Armstrong@dtaa.com
27715 0 operose assistance unrelenting demanding job lets talk today
```

```
http {U4O1-O1IE61HB-3238USSY} 9/30/2010 16:02 BSS0369 PC-3672
http://www.relytec.com/Climate_of_Florida/suniest/NNXrlybttre1893061997.aspx
surveillance download download download keyboard hidden file everything
device {P0D1-D3EP98MZ-5216UZLY} 9/30/2010 16:07 BSS0369 PC-3672
Connect
file 9/30/2010 16:10 BSS0369 TBDZA9ME.exe 4D-5A-90
keyboard program username monitor download password captured
device {F0B5-B3GP27FZ-1059KQUY} 9/30/2010 16:15 BSS0369 PC-3672
Disconnect
logon {O4B2-O6CO48UA-3416FAAM} 9/30/2010 20:16 BSS0369 PC-8486
Logon
device {S6U3-G1GM61UU-6821UTNG} 9/30/2010 20:26 BSS0369 PC-8486
Connect
device {L5V0-T7KT12VV-5753XJEO} 9/30/2010 20:30 BSS0369 PC-8486
Disconnect
logon {F3A4-G8RG38TH-4582NBDR} 9/30/2010 20:31 BSS0369 PC-8486
Logoff
logon {F5U1-K4NZ89MG-2416CBXD} 10/1/2010 19:19 BSS0369 PC-8486
Logon
logon {C7V6-T5HQ75BY-8395TCFN} 10/1/2010 19:33 BSS0369 PC-8486
Logoff
logon {F2X9-N6CT67WV-0165DWKQ} 10/1/2010 19:40 FBA0348 PC-8486
Logon
email {A8K0-Q3DX55DM-3791QCFI} 10/1/2010 19:45 FBA0348 PC-8486
Arthur.John.Floyd@dtaa.com Francis.Brian.Armstrong@dtaa.com 47278
logon {B1I2-X3TE24CB-4083EMHJ} 10/1/2010 19:48 FBA0348 PC-8486
Logoff
```

Now, the corresponding contextual data for above log is:

Frances Alisa Wiggins exhibited a drastic shift in behavior starting on July 22, 2010, marked by a series of alarming actions that raised significant concerns within the organization. Initially, Wiggins began to send out a total of 13 emails to various colleagues, which included both routine inquiries and potentially sensitive information. Notably, these communications contained a tone of urgency and discontent, hinting at deeper issues within the workplace. The sudden increase in email activity contrasted sharply with Wiggins's previous patterns, suggesting a possible attempt to manipulate or instigate responses from her peers. On the same day, Wiggins downloaded a keylogger onto a supervisor's machine, demonstrating a clear intent to engage in malicious activities. This action, coupled with the fact that she had no prior authorization to access such software, highlights a significant breach of trust and company policy. The keylogger was likely used to capture sensitive information, including usernames and passwords, which could have been misused for personal gain or to compromise organizational security. Wiggins's behavior escalated further as she began to use a thumb drive to transfer files to a removable drive, a move that raised red flags regarding potential data theft or sabotage. Additionally, there were reports of Wiggins accessing sensitive company data, including confidential documents and financial records, which further underscored the severity of the situation. On July 23, 2010, Wiggins logged in to a supervisor's account, exploiting the trust placed in her to manage sensitive information. This unauthorized access not only demonstrated a blatant disregard for security protocols but also highlighted a potential vulnerability within the organization's IT infrastructure. The culmination of these actions culminated in Wiggins sending out a mass email to the entire organization, containing a budget notice that was both alarming and potentially damaging. This move was seen as an attempt to instigate panic and distrust among colleagues, further destabilizing an already precarious work environment. In summary, Wiggins's actions between July 22 and 23, 2010, exhibited a concerning pattern of malicious intent, marked by the use of a keylogger, unauthorized access to sensitive information, and the dissemination of alarming communications. These actions not only compromised organizational security but also demonstrated a clear disregard for the welfare of colleagues, raising significant concerns regarding Wiggins's motives and integrity within the workplace.

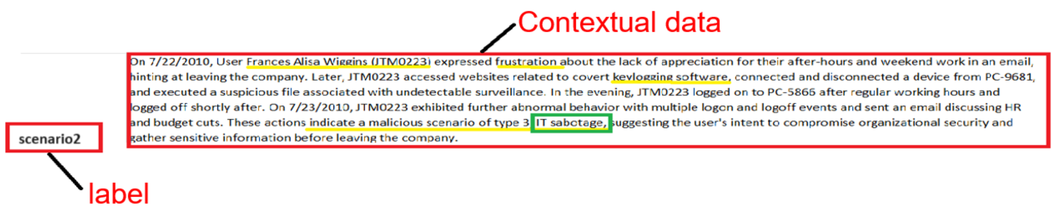
The table below shows data generated from the large language model that is, fine-tuned Llama3.1. It has generated textual data with less than 500 words, the length of textual data has been taken into account as SecureBERT takes input of maximum 512 tokens. The final dataset used to train the SecureBERT model for classification purpose consisted of both simulated and synthetic dataset. 40% of the contextual log data was generated from the Llama 3.1 model and 60% of the data was synthesized for the training purpose.

Table 4.1: Sample of generated contextual log data with label

text	**label**
On July 25, 2024, at 16:20, a user with the us...	scenario1
On July 28, 2024, at 11:13 AM, username KYKP45...	normal
On July 5, 2024, at precisely 3:06 AM, an acco...	scenario3
On July 20, 2024, BVLD4612 logged into their w...	normal
On July 23, 2012, at 20:13, user WITB9761 exhi...	scenario2
...	...
On July 11, 2024, at precisely 05:19, an alarm...	scenario3
On July 24, 2012, an employee using the userna...	scenario2
On July 27, 2024, from 05:52 to 13:52, an inci...	scenario3
On July 30, 2024, at precisely 16:39, an incid...	scenario3
Username MINP1179 logged into the system at 01...	normal

The above shown data are generated separately as labeled as scenario1, scenario2, scenario3 and normal. After that, all the data are shuffled by using pandas library.

Here is a more detailed explaination about the data generated:


Contextual data
scenario2
label

On 7/22/2010, User Frances Alisa Wiggins (JTM0223) expressed frustration about the lack of appreciation for their after-hours and weekend work in an email, hinting at leaving the company. Later, JTM0223 accessed websites related to covert keylogging software, connected and disconnected a device from PC-9681, and executed a suspicious file associated with undetectable surveillance. In the evening, JTM0223 logged on to PC-5866 after regular working hours and logged off shortly after. On 7/23/2010, JTM0223 exhibited further abnormal behavior with multiple logon and logoff events and sent an email discussing HR and budget cuts. These actions indicate a malicious scenario of type 3 IT sabotage, suggesting the user's intent to compromise organizational security and gather sensitive information before leaving the company.

Figure 4.2: An example of generated contextual data

4.3 Training SecureBERT for Classification

The histogram representing data distribution according to label can be shown in the figure below:

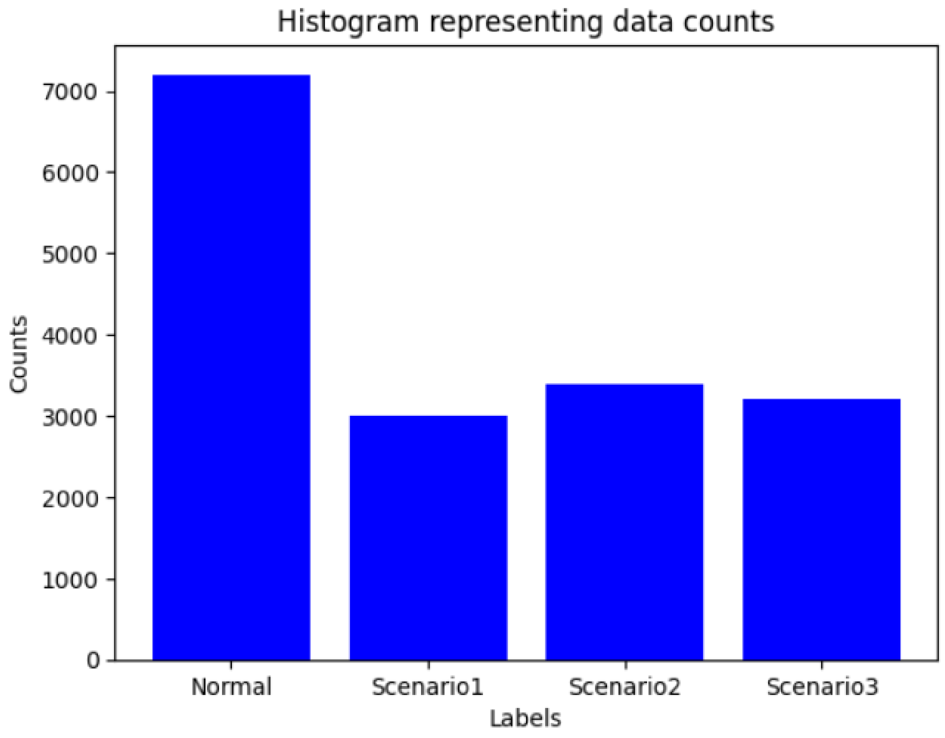


Figure 4.3: Data distribution for training SecureBERT

The distribution of data counts across the four categories of ”Normal,” ”Scenario 1,” ”Scenario 2,” and ”Scenario 3” is depicted in the histogram. With over 7000 data points, the ”normal” category has the most, with the remaining three categories (Scenario 1, Scenario 2, and Scenario 3) having counts that are very similar, with each category having between 3000 and 3500 data points. The ”normal” group has more than double the counts of the other categories, highlighting the notable difference in the data distribution.

The secureBERT model when trained by adding a classification head for multi class classification, the samples size was divided into validation, test and train set. The percentage of splitting of values for each set can be shown in the piechart below:

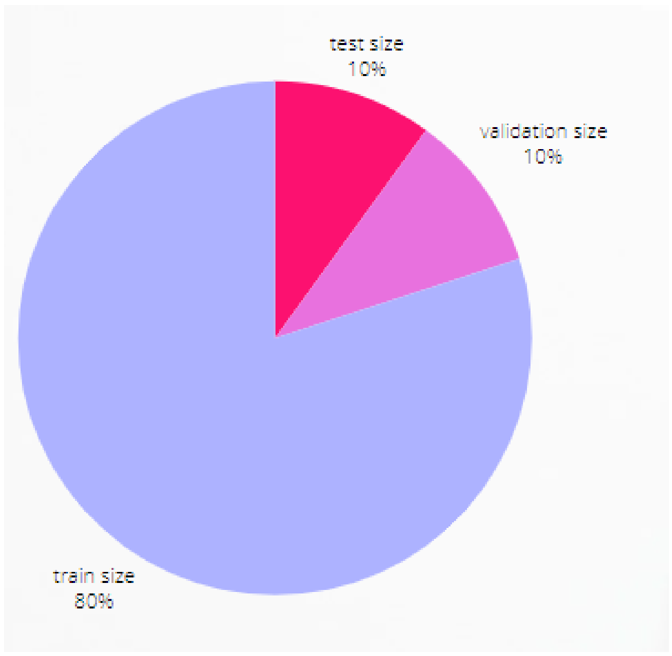


Figure 4.4: Dataset distribution in training, testing and validation for training SecureBERT

When trained the model for binary classification, with learning rate = 1e-5, batch size = 8 and epochs 25. Early stopping criteria was also implemented with patience 3.

Table 4.2: Hyperparameters for Training

Hyperparameter	**Value**
Epochs	25
Learning Rate	1e-5
Batch Size	8
Max Length	512
Patience	3
Weight Decay	0.01

The results obtained for this combination is shown below:

The training and validation loss curve obtained for the above combination of hyperparameters can be seen in the fugres given below:

Table 4.3: Train results

Epoch	Train Loss	Train Acc	Val Loss	Val Acc	Prec.	Recall	F1-score
1/25	0.8456	0.6781	0.3083	0.9750	0.9423	0.9516	0.9406
2/25	0.2178	0.9474	0.0719	0.9958	0.9934	0.9919	0.9926
3/25	0.0487	0.9969	0.0054	1.0000	1.0000	1.0000	1.0000
4/25	0.0092	1.0000	0.0017	1.0000	1.0000	1.0000	1.0000
5/25	0.0034	1.0000	0.0009	1.0000	1.0000	1.0000	1.0000
6/25	0.0022	1.0000	0.0006	1.0000	1.0000	1.0000	1.0000
7/25	0.0014	1.0000	0.0003	1.0000	1.0000	1.0000	1.0000
8/25	0.0010	1.0000	0.0003	1.0000	1.0000	1.0000	1.0000
9/25	0.0008	1.0000	0.0003	1.0000	1.0000	1.0000	1.0000

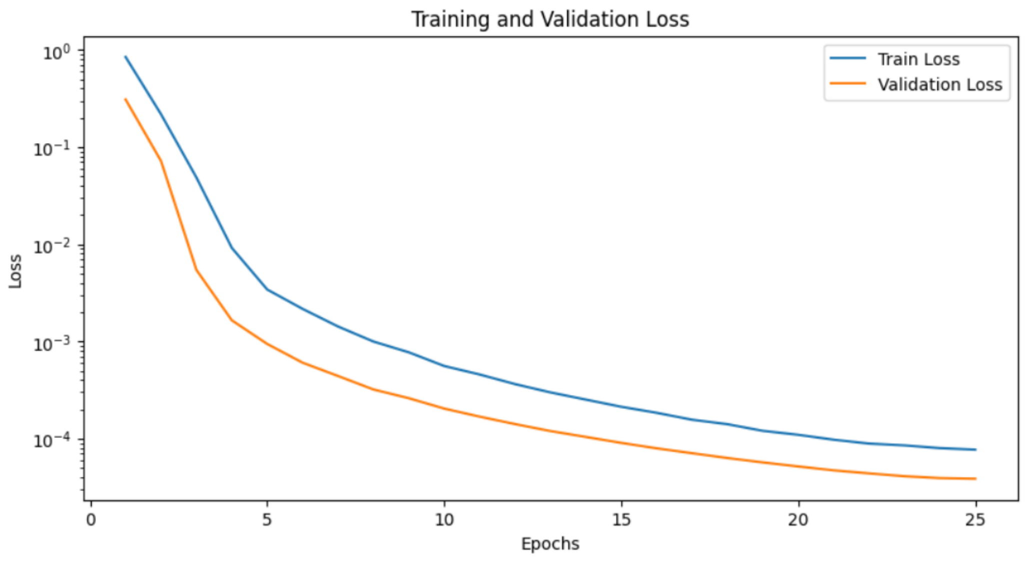


Figure 4.5: Training and validation loss curve for classification

The figure above depicts the training and validation loss curves over 25 epochs of a deep learning model. The x-axis represents the number of epochs, while the y-axis represents the loss values on a logarithmic scale. Two curves are plotted: the training loss (in blue) and the validation loss (in orange).

Throughout the training process, both the training and validation losses decrease consistently, indicating that the model is learning effectively. The initial rapid decline in the loss values suggests that the model quickly captures significant patterns in the data. As the epochs progress, the rate of decrease slows down, indicating that the model is gradually converging towards its optimal performance.

The relatively smooth and parallel behavior of the training and validation loss curves suggests that the model is generalizing well to unseen data, without significant overfitting.

The validation loss does not exhibit a notable increase or divergence from the training loss, which is a positive indicator of the model's stability and robustness. This behavior reflects a successful training process, where the model maintains good generalization while minimizing the loss on both the training and validation sets.
Hyperparameter tuning using the GridSearchCV and was performed on the same model along with L2 regularization technique and AdamW optimizer. The hyperparameter grid used for GridSearchCV is shown in the following table:
After grid search analysis, the best hyperparameters obtained are:

Table 4.4: Hyperparameter Grid for GridSearchCV

Parameter	Values
learning_rate	[1e-5, 5e-5, 1e-4, 2e-5]
batch_size	[8, 16, 32]
epochs	[3, 4, 5, 10, 20,25]

- learning rate = 1e-5
- batch size = 16
- number of epochs = 25

The model was trained using the best hyperparameters obtained from the grid analysis by GridSearchCV and as shown in the Figure4.5, training and validation loss curve was obtained smooth that does not show any overfitting of the model and ensures the model is learning well and can perform accurately in the classification task.
The training and validation accuracy curve is shown as:
The model's effectiveness in picking up the necessary skills in a short amount of epochs is demonstrated by its capacity to attain such high accuracy, which can drastically cut down on the amount of time and computing power required for training. The model appears to be both accurate and well-calibrated, indicating that it can provide predictions on new data with confidence based on the near alignment of the training and validation accuracy. The model's good performance on both datasets increases its dependability and positions it as a viable option for use in practical applications.

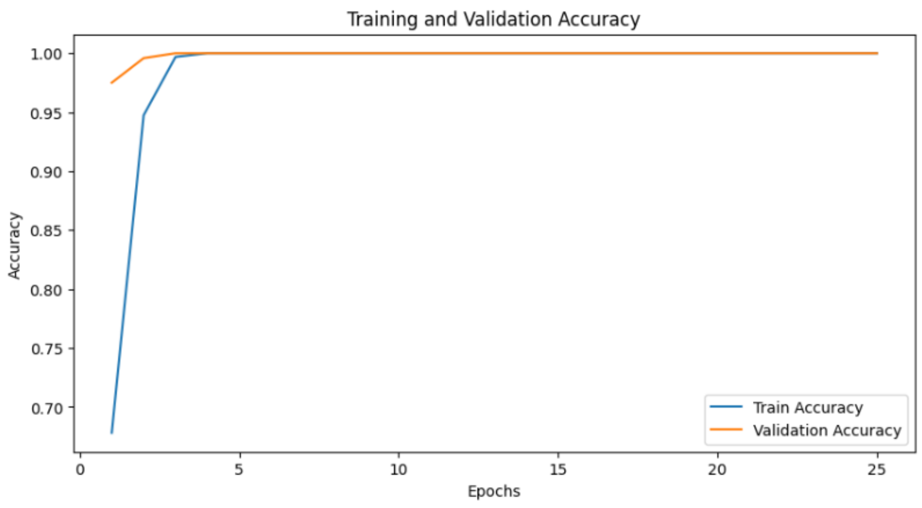


Figure 4.6: Training vs Validation Accuracy curve

In addition, the model's stability across multiple epochs suggests that it has reached a stable and consistent learning level, which lowers the probability of performance variations. Maintaining confidence in the model's predictions in a variety of settings depends on this consistency. Overall, the figure highlights the model's impressive capability to generalise well while maintaining high accuracy, ensuring it will likely perform well in practical use cases.

Now the trained model was tested on different sets of unseen data for the evaluation of the model, which is dicussed in the next section.

4.4 Testing on different sets of Test Data

The trained model was tested on different test data sets, including the splitted set along with the newly generated data for testing purpose. These are discussed here.

4.4.1 Testing on splitted testing data

Firstly, the model was evaluated on the splitted test data, which was obtained by splitting the dataset into training, testing and validation set. It was seen that the model accurately classifies in this dataset. The confusion matrix shows the performance of the model when evaluated on the test set data across four different classes (Class 0, Class 1, Class 2, and Class 3). The matrix indicates that the model perfectly predicted all instances for each class, with no misclassifications. For Class 0, all 152 instances were correctly

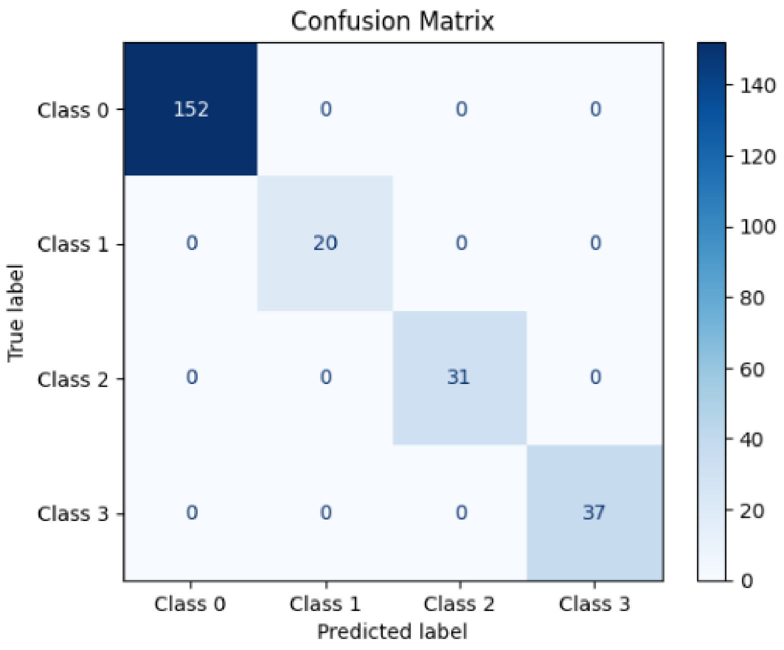


Figure 4.7: Obtained Confusion matrix when model tested on testing set data

classified, while Class 1 had 20 correct predictions, Class 2 had 31, and Class 3 had 37. The absence of off-diagonal values (non-zero entries outside the diagonal) indicates that the model made no errors, meaning that each test instance was classified into its true class without confusion. This highlights the model's exceptional accuracy and effectiveness in distinguishing between the different classes in the test data.

4.4.2 Testing on 100 unseen test data

The trained model was now tested on 100 newly generated unseen test data. There were the new dataset which was not used in training or validation. The confusion matrix and roc curve obtained after the particular experiment are discussed below:

Confusion Matrix

The model was given the task of classifying data into four categories: ”scenario 1,” ”scenario 2,” ”scenario 3,” and ”normal.” The performance of the model on this test set is displayed in this confusion matrix. The top-left cell of the matrix indicates that all 39 occurrences of ”scenario 1” were correctly identified by the model. On the other hand, 18 cases of ”scenario 2” were correctly classified, while 3 instances of ”scenario 3” were wrongly classified. For ”scenario 3,” every one of the twenty cases was appropriately classified. 14 occurrences of the ”normal” class were accurately detected

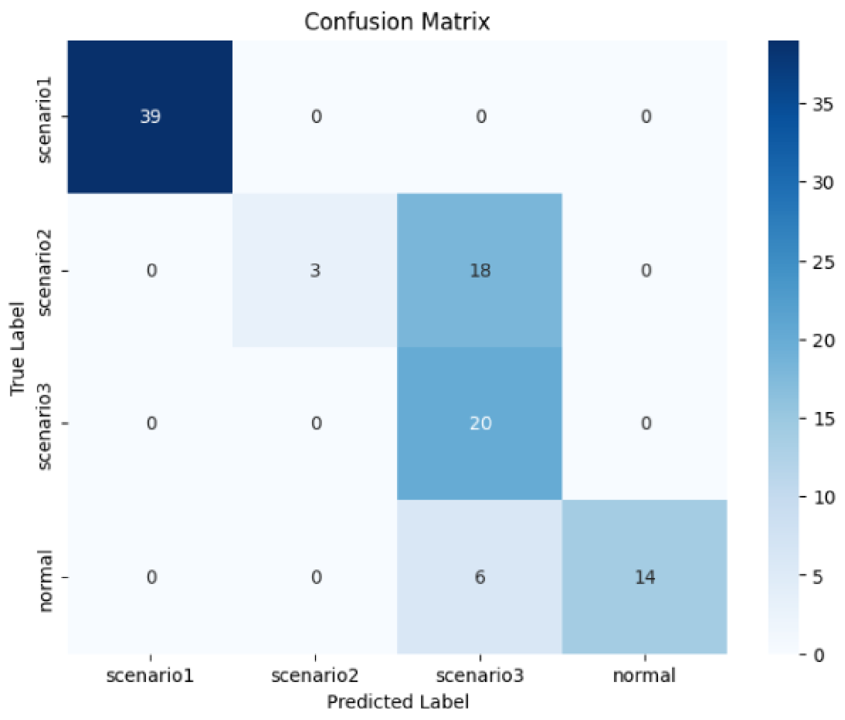


Figure 4.8: Obtained confusion matrix on 100 unseen test data

by the model; however, 6 cases were incorrectly classified as ”scenario 3.” While there is some confusion between ”scenario2” and ”scenario3,” as well as some misclassifications between ”normal” and ”scenario3,” the model performs well overall, especially for ”scenario1” and ”scenario3.” Despite these errors, the model performs well in accurately identifying the majority of instances in the test set.

Roc Curve

The ROC curve shows the model’s classification performance for each of the four classes: normal, scenario 1, scenario 2, and scenario 3. The trade-off between the true positive rate (sensitivity) and the false positive rate (specificity) is illustrated by the curves for each class. The area under the curve (AUC) of 1.0 on the ROC curves for the normal class and scenario 1 indicates perfect classification. The AUC values for scenarios 2 and 3 are marginally lower (0.996986 and 0.990000, respectively), suggesting small misclassifications. The confusion matrix clearly shows these misclassifications: six samples of normal are misclassified as scenario 3, and three samples of scenario 2 are misclassified as scenario 3. The model works well in scenarios 1 and normal, but it has some difficulty in scenarios 2 and 3.

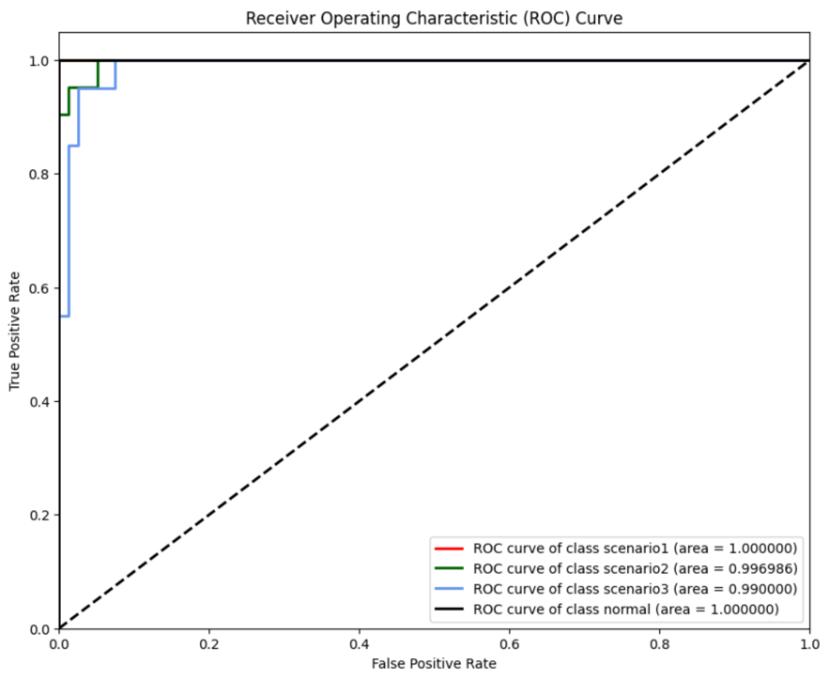


Figure 4.9: Roc curve for 100 unseen test data

4.4.3 Testing on 1200 unseen test data

The trained model was now tested on 1200 newly generated unseen test data. There were the new dataset which was not used in training or validation. The confusion matrix and roc curve obtained after the particular experiment are discussed below:

Confusion matrix

The presented confusion matrix offers a clear assessment of the performance of the classification model. Four classes make up the matrix: ”scenario1,” ”scenario2,” ”scenario3,” and ”normal.” In comparison to the actual class label, each cell in the matrix indicates the number of cases that the model predicted to belong to a particular class. Correct classifications are shown by the diagonal elements when the true and predicted labels coincide. As an example, the model identified all 300 occurrences of ”scenario 1,” ”scenario 2,” and ”scenario 3” properly and without making any mistakes. On the other hand, out of 300 occurrences for the ”normal” class, 285 were correctly identified as ”scenario 3,” while 15 were mistakenly labelled. This shows that ”scenario 1,” ”scenario 2, and ”scenario 3” have high accuracy rates, whereas the ”normal” class has a low mistake rate.

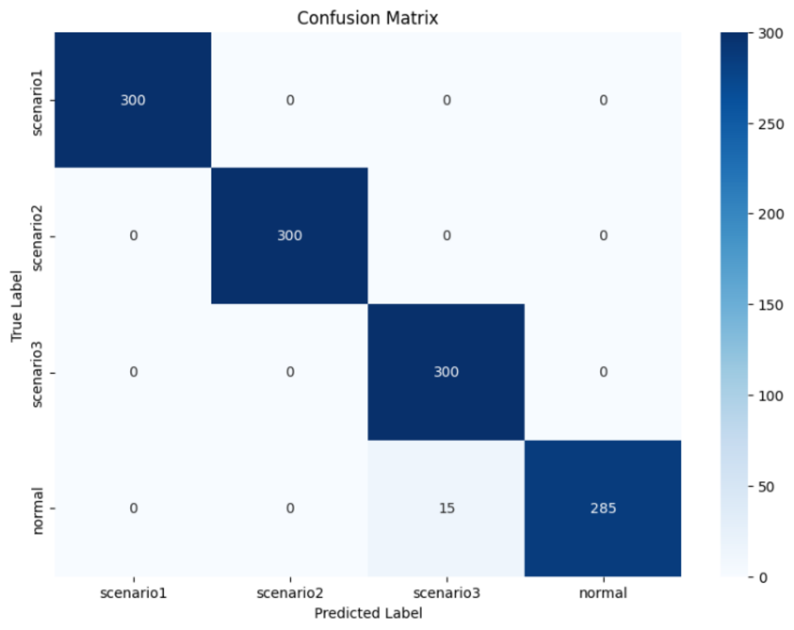


Figure 4.10: Obtained confusion matrix for 1200 unseen test data

Roc Curve

The Receiver Operating Characteristic (ROC) curve visualizes the performance of a multi-class classification model. Each line in the graph represents one class, plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The Area Under the Curve (AUC) is a metric that summarizes the model's ability to distinguish between classes. In this case, the AUC values are close to 1 for all classes (Class 0, Class 1, Class 2, and Class 3), indicating that the model has excellent discriminatory power. The near-perfect ROC curves suggest that the model performs exceptionally well, almost perfectly distinguishing between the classes with minimal false positives and false negatives. The slight deviation for Class 2 (with an AUC of 0.999937) suggests an almost negligible error rate.

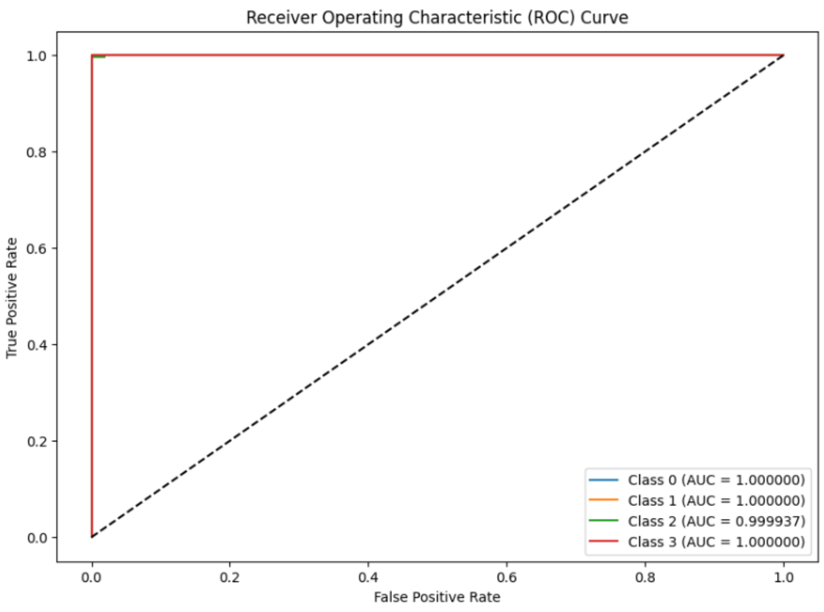


Figure 4.11: ROC curve for 1200 unseen test data

5 DISCUSSION AND ANALYSIS

5.1 Comparison of Theoretical and Simulated Outputs

5.1.1 Theoretical Expectations

Our theoretical framework shows that increasing the dataset size and number of epochs would enhance the performance of both the large language models that is, the context generation and the classification model.This expectation stemmed from the idea that bigger datasets offer a wider range of instances, which promotes deeper learning. Furthermore, it was predicted that methods such as early halting, callbacks for learning rate reduction on plateau, and learning rate reduction would assist fine-tune the model, reducing overfitting and enhancing generalisation.

5.1.2 Simulated Outputs of Context Generation model

Our experiments' outcomes substantially supported these predictions. In first experiment, the Llama 3.1 model was fine-tuned using 240 data only, which was synthetically generated by the OpenAI's API using the **"gpt4omini"** model. However, during this experiment, the output was not satisfactory enough to train the secureBERT model for classification.

In the second experiment, the Llama3.1 model was configured with parameter efficient fine-tuning using the Low-rank adaptation technique along with the flash attention mechanism by using X-formers. With this configuration, the Llama3.1 model was fine-tuned with 60 epochs and the learning curve showed the model learned pretty well with adjusting the weights correctly. At this point, the dataset was increased to 1400 samples containing over 300 samples for each scenarios. At the end of this experiment, the results were quite explanatory and contained the key information that are required to classify the user behavior between normal or abnormal. Here, abnormal includes scenario 1, scenario 2 and scenario 3. Here, is an example of contextual data generated by the fine-tuned Llama model for scenario 1.

"On the evening of September 30, 2010, at 13:31, a concerning shift in behavior began to unfold for an employee identified as Brenden Samuel Shaffer. Over the course of the next six hours, he engaged in a series of alarming activities that starkly contrasted with his previous work patterns. Initially, he sent a total of 16 emails to his supervisor, which included 13 messages that were flagged as urgent, indicating a sudden increase in his communication frequency. This change in behavior was particularly notable given that in the past, Brenden had never utilized the urgent flag before. As the night progressed, his actions became even more concerning, as he began downloading a significant number of files, including a suspicious piece of software that raised red flags for potential malicious intent. During this time, he also accessed his supervisor's account without authorization, a clear violation of company policy. The combination of these actions suggested a calculated effort to manipulate information and potentially undermine organizational trust. In a further alarming development, Brenden connected a removable drive to his workstation, which could indicate a possible attempt to exfiltrate sensitive data from the company. This act, coupled with the previous unauthorized access, escalated the severity of the situation. On October 1, 2010, at 19:19, Brenden logged in for the first time, marking the beginning of a new workday that would ultimately lead to his departure from the organization. Throughout this session, he sent 15 emails to various colleagues, which included a few that contained misleading information, further contributing to the chaos. The culmination of these actions took place when Brenden sent an email to the entire organization, which included a mass resignation of funding and intention to down size, effectively causing a significant impact on the team's morale and productivity. As the night wore on, Brenden logged in a total of 14 times, indicating a clear pattern of after-hours activity that deviated sharply from his typical work habits. On the following day, October 2, 2010, he left the organization without notice, leaving behind a trail of confusion and concern among his colleagues and management. The activities exhibited by Brenden Samuel Shaffer from 13:31 on September 30, 2010, to 19:19 on October 1, 2010, paint a concerning picture of potential insider threat behavior. The combination of urgent emails, unauthorized access, removable drive usage, and the sudden departure from the organization raise significant alarms regarding the security and integrity of the company's data and operational processes."

This model can generate the contextual data maximum upto 500 words. This has taken into account as the secureBERT process the texts of the maximal length upto 512 tokens.

5.2 Error Analysis

5.2.1 Sources of Errors

1. Data Quality

    - Noisy Data: The initial training data contained mislabeled examples or irrelevant features,that caused confusing the model during training, leading to poor generalization.
    - Insufficient Data: A lack of sufficient training examples for each scenario resulted in overfitting or poor performance due to the model not learning the underlying patterns effectively.

2. Model Architectures and Hyperparameters

    - Inappropriate Model Size: Fine-tuning a model that is either too large for the dataset lead to overfitting and underfitting, respectively.
    - Learning Rate Issues: Setting the learning rate too high caused the model to converge too quickly to a suboptimal solution, while too low a learning rate lead to slow convergence or getting stuck in local minima.
    - Batch Size and Gradient Accumulation: Incorrect batch size or gradient accumulation steps impacted training stability and model performance.

3. Contextual Understanding

    - Loss of Contextual Understanding: During fine-tuning, the model does not adequately capture the nuances of different scenarios, which struggle to differentiate between them, leading to errors in classification.
    - Inadequate Tokenization: The tokenizer used was not suited to the specific language or domain of the data, important contextual information was lost, affecting the model's ability to generate accurate predictions.

5.2.2 Error Analysis and Impact

- Data Quality: The data quality was improved increasing the number of samples of each class and including key informations regarding those labels. This was done by changing the prompt during synthetic data generation for fine-tuning the llama model. Also, the quality of data and number of samples for each of the class was increased for training securebert model as well.

- Contextual Understanding: Grouped Query Attention (GQA) tokenization technique was used during fine-tuning llama model and wordpiece subword tokenizer technique was used for training the secureBERT model for proper contextual understanding.

- Model Architecture and Hyperparameters: Llama model was loaded in 4 bit quantization and configured with parameter efficient fine tuning technique and hyperparameters for classification model was set to the optimal values obtained by performing grid analysis.

5.3 Best Case Scenario

The model performed best for learning rate 1e-5 with batch size of 16 and training epochs of 25 for the classification that is, normal and abnormal classification of the data. Here, the plot shows training and validation loss curve upto 5th epoch, it follows the similar pattern afterwards.

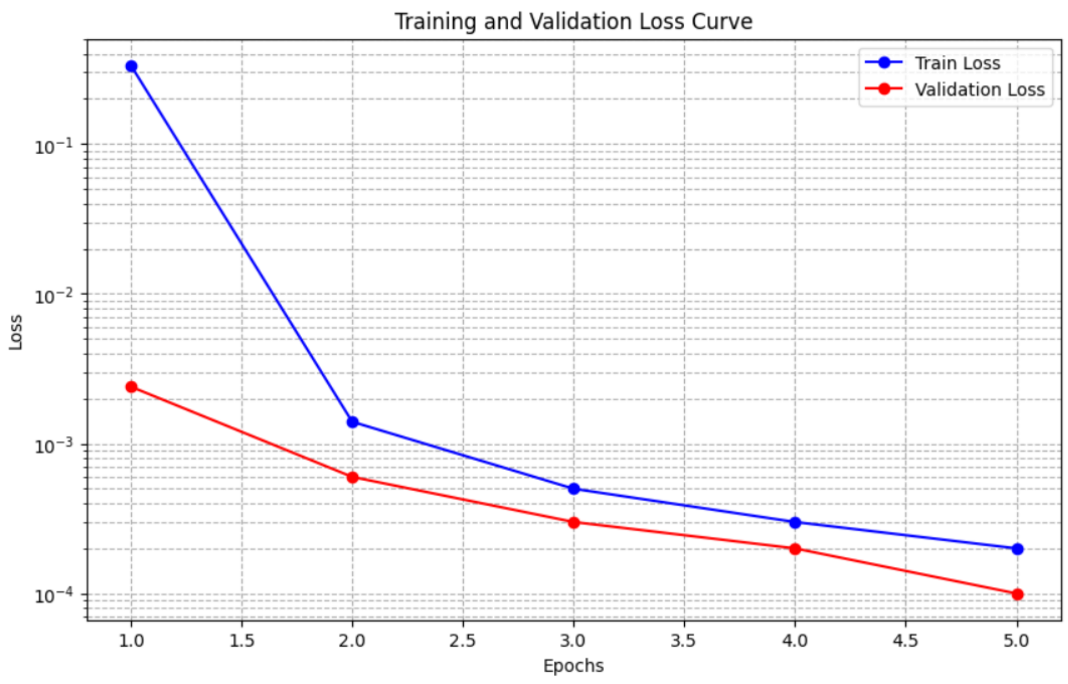


Figure 5.1: Training and validation loss curve for classification

In the above curve, it can be observed that these parameters can obtain the good results showing more gap between training and validation loss during starting of the epoch. As the epoch increases the difference can be seen as decreasing, which is good for a model.

5.4 Worst Case Scenario

The same model was trained with epoch 40 with learning rate lr = 5e-6 and batch size of 8. Here, the model seems to be overfitting after 14 epoch.

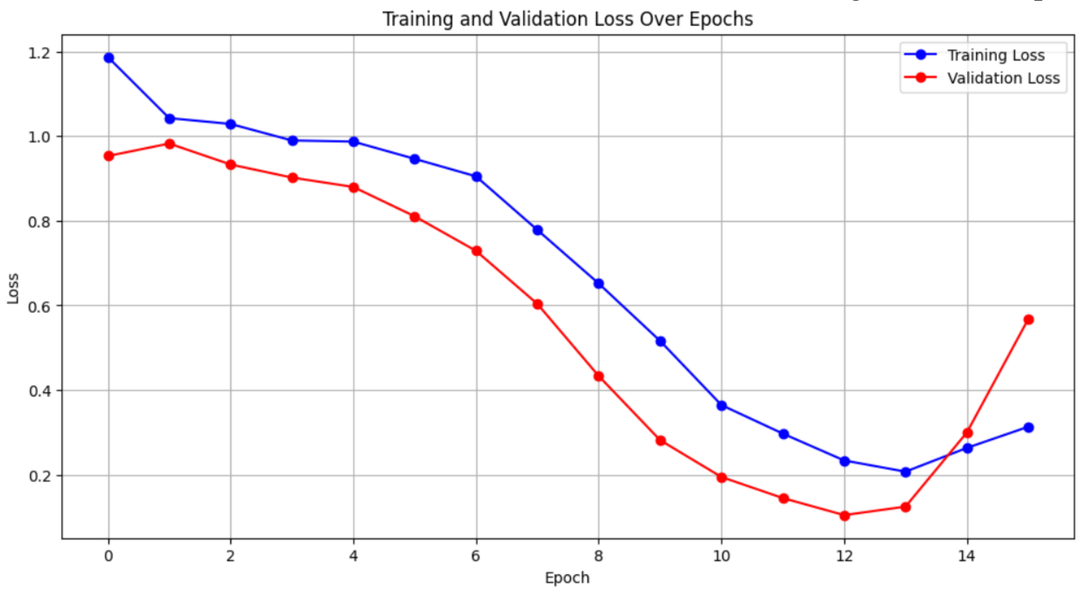


Figure 5.2: Training and validation loss curve for multiclass classification

In the above curve, it can be observed that these parameters can obtain the good results

showing more gap between training and validation loss during starting of the epoch. As the epoch increases the difference can be seen as decreasing till 14th epoch. After that, it seems to be overtfitting the model.
From the results, it can be concluded that the model efficiency can be increased by using more data samples and better combination of the hyper parameters.

5.5 Second Experiment

In the second experiment, SecureBERT was trained on a diverse set of over 2400 datasets with varying hyperparameters, aiming to optimize its performance on the specific task. The learning rate was set at 1e-5, which is relatively low, indicating a cautious approach to weight updates that helps in preventing the model from overshooting minima during optimization. The batch size was 16, a choice that balances the trade-off between computational efficiency and the stability of the gradient estimates. Training was conducted over 5 epochs, which is generally sufficient for models of this scale to converge, especially when fine-tuning on a large number of datasets.
The model achieved a 76% accuracy on the test data, which reflects the effectiveness of the chosen hyperparameters and the robustness of the training process. However, this performance suggests there is still room for improvement, potentially by further tuning the hyperparameters or increasing the model complexity. The result highlights the importance of carefully balancing the learning rate and batch size to achieve optimal performance, as well as the significance of training duration in ensuring the model can adequately learn from the data without overfitting.

5.6 Experiment with increased dataset

In another experiment, SecureBERT was trained for a multiclass classification task on a substantially larger dataset of over 16,000 samples. The training process utilized a learning rate of 2e-5, which is slightly higher than in the previous experiment, allowing the model to adjust its weights more aggressively. The training was carried out over 25 epochs with a batch size of 16, a consistent choice that maintains stability in gradient estimation while processing a manageable amount of data per iteration. Additionally, weight decay was set to 0.01, introducing regularization to prevent overfitting by penalizing large weights.

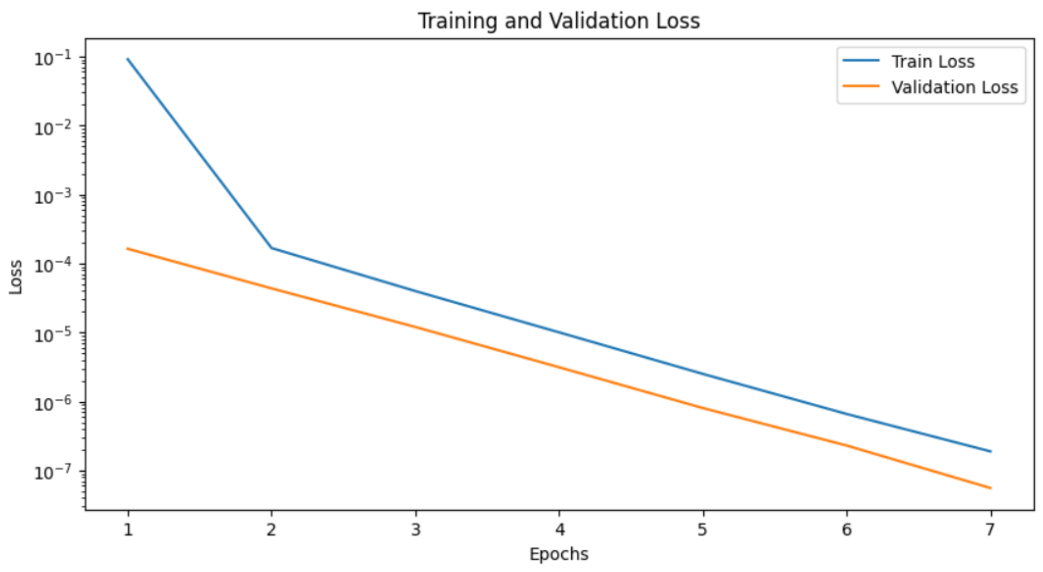


Figure 5.3: Training and validation loss curve for classification on increased dataset

The model's performance during training was highly impressive, starting with a 94% accuracy in the first epoch and quickly reaching 100% by the third epoch. This rapid increase in accuracy suggests that the model was able to learn the underlying patterns in the data very effectively, likely due to the larger dataset and the refined hyperparameters. However, the early achievement of perfect training accuracy also raises concerns about potential overfitting, especially given the relatively high number of epochs. This suggests that while the model has learned the training data exceptionally well, its generalization to unseen data would need to be carefully evaluated to ensure it hasn't simply memorized the training set. Despite the impressive training performance, the model was subjected to early stopping after 7 epochs, with a patience parameter of 5. This means that if the model's performance on a validation set did not improve for 5 consecutive epochs, the training was halted to prevent overfitting. Early stopping was triggered relatively early in the training process, despite the model initially achieving 100% accuracy on the training data by the third epoch.

The fact that the model was stopped at the 7th epoch indicates that while it quickly learned the training data, further training did not result in significant improvements on the validation set. This decision to use early stopping was crucial in ensuring that the model didn't overfit, as continuing training could have led to diminished generalization

ability on unseen data. The early stopping at the 7th epoch, despite the high training accuracy, underscores the importance of monitoring validation performance to balance learning and prevent the model from becoming overly specialized to the training data.
To conclude, The experiments conducted with SecureBERT highlight the delicate balance required in model training, particularly in handling large datasets and optimizing hyperparameters for robust performance. In the first experiment, the model achieved a respectable 76% accuracy on the test data, demonstrating effective learning, though with potential for further improvement. The second experiment showcased SecureBERT's ability to quickly achieve near-perfect accuracy on the training data, aided by a well-chosen learning rate, batch size, and weight decay. However, the early stopping mechanism, triggered after 7 epochs, was crucial in preventing overfitting, indicating that the model's rapid learning plateaued in terms of generalization.
These findings emphasize the importance of not just achieving high training accuracy but also maintaining model generalization. Early stopping proved to be a valuable strategy in managing overfitting, ensuring that the model remains effective on unseen data. Moving forward, further experiments could explore additional regularization techniques or adjustments to the learning rate schedule to enhance the model's performance on validation and test datasets.

6 FUTURE ENHANCEMENTS

While the primary objectives of this project have been successfully met, minor enhancements can be pursued to further improve the performance in context generation model and usability of the developed models. These future enhancements focus on refining the models for better accuracy, expanding their applicability, and ensuring their readiness for real-world deployment.

6.1 Evaluation on Broader Datasets

One of the critical future enhancements involves the evaluation of both the LLaMA 3.1 and SecureBERT models on a wider range of datasets. Although the models performed well in the initial experiments, their effectiveness needs to be tested on datasets that cover a broader spectrum of security domains and real-world scenarios. Cross-domain testing can help determine the versatility of the models when applied to different types of data, such as network traffic, logs from intrusion detection systems, or application-level data. This expanded testing will be crucial in assessing how well the models generalize beyond the specific datasets on which they were initially trained.

Additionally, testing the models on real-world datasets will validate their performance under more practical conditions. The inclusion of data that reflects actual malicious activities and normal behavior in real environments will allow for a more accurate assessment of their effectiveness in operational security scenarios. For instance, incorporating datasets from diverse industries such as healthcare, finance, and telecommunications will help ensure that the models are not only theoretically robust but also practically applicable across various sectors. This evaluation will serve as a litmus test for the models' generalizability and adaptability.

Furthermore, applying k-fold cross-validation during these evaluations will provide a deeper understanding of the models' performance consistency. Cross-validation will reveal how well the models generalize across different subsets of data, highlighting any overfitting issues and areas that require further training. The insights gained from this broader evaluation will guide future refinements and help in selecting the best configurations for real-world deployments.

6.2 Optimization of LLaMA 3.1 for More Complex Language Scenarios

While the LLaMA 3.1 model has shown promising results in generating contextual data from structured inputs, it can benefit from further optimization, particularly for handling more complex language scenarios. The current model performs well on straightforward security descriptions but may struggle with generating nuanced explanations or interpreting more sophisticated input data. Future work should focus on expanding the training dataset to include more complex linguistic structures and domain-specific terminologies. This will enhance the model's ability to generate accurate, coherent, and contextually appropriate natural language outputs in diverse cybersecurity situations.

Incorporating Reinforcement Learning from Human Feedback (RLHF) into the optimization process could also be an effective strategy. By continuously learning from feedback provided by human experts, LLaMA 3.1 can be trained to produce more refined and contextually relevant outputs. For example, security analysts could provide feedback on the quality and accuracy of the generated contextual data, which the model can use to iteratively improve its performance. This dynamic learning approach will be particularly useful for adapting the model to evolving security threats and increasingly complex language scenarios that may arise in real-world applications.

Another area of optimization involves improving the model's interpretability when dealing with ambiguous or incomplete data. Enhancing its ability to infer meaning from partially structured inputs will allow it to provide more insightful and detailed contextual data, even in situations where the data is sparse or incomplete. This will be particularly valuable in cybersecurity, where incomplete logs or fragmented data often need to be interpreted to provide a coherent narrative.

6.3 Real-Time Testing and Deployment

Finally, real-time testing and deployment of the models will be a crucial step in transitioning them from a research environment to practical, operational use. Deploying the models in a real-world setting, such as a Security Operations Center (SOC), will allow for the assessment of their performance in real-time scenarios where speed, accuracy, and reliability are critical. The development of a robust inference pipeline will be necessary to ensure that the models can handle incoming data streams efficiently and respond with actionable insights in a timely manner. The success of real-time testing will determine the readiness of the models for deployment in mission-critical environments.

Real-time testing will also expose the models to new challenges, such as latency, data variability, and unexpected input formats, which are not typically encountered during controlled, offline testing. These challenges will provide valuable feedback for further refining the models. For instance, tuning the models to reduce inference times without sacrificing accuracy will be essential in ensuring their viability for deployment in environments where rapid decision-making is required, such as in intrusion detection or automated threat response systems.

7 CONCLUSION

This initiative aimed to solve the following research question: **How can large language model be effectively used to convert structured data into natural language contextual data, and how can BERT models be optimized to accurately classify between normal and malicious data in insider threat scenarios?** The project addressed these two objectives by leveraging state-of-the-art models in natural language processing (NLP) and classification, specifically focusing on fine-tuning LLaMA 3.1 for contextual data generation and training SecureBERT for multi-class classification of security events. The key supporting ideas discussed throughout the project centered on the integration of NLP and security classification. First, the successful fine-tuning of LLaMA 3.1 enabled the conversion of structured cybersecurity data into natural language outputs that are coherent and contextually accurate. This achievement is essential in bridging the gap between complex security information and human understanding, particularly for analysts who rely on textual explanations to make informed decisions. Secondly, SecureBERT was trained and optimized to classify security events into multiple categories, including normal behavior, scenario1, scenario2, and scenario3. The classification performance of SecureBERT demonstrated an improvement over traditional deep learning models, particularly in scenarios where fine distinctions between different types of malicious behaviors were required. Lastly, the project identified areas for future enhancements, including optimization of the models through cross-validation on broader datasets, and real-time deployment testing to ensure scalability and robustness.

This project has provided its contribution to the area of **cybersecurity and natural language processing** in the given ways in order to achieve the goal:

- **Contextual Data Generation**: Successfully fine-tuning the LLaMA 3.1 model allowed for the generation of natural language explanations from structured cybersecurity data, making complex information more interpretable.
- **Multi-Class Classification**: Training the SecureBERT model enabled accurate classification of various security scenarios(99-100%), including normal and malicious behaviors, with better performance than traditional deep learning models.
- **Performance and Scalability**: The project's results showed improvements in accuracy

and performance, but further optimization was suggested to enhance the models' real-time applicability and to ensure they can handle larger, more diverse datasets efficiently. With the above stated contributions, this project has shown the hopes for the effective integration of NLP and machine learning in cybersecurity. By translating structured data into understandable natural language, the models enhance the interpretability of security incidents. Simultaneously, the classification capabilities of SecureBERT offer reliable and robust detection of malicious activities. These advancements represent a significant step forward in automating security operations while maintaining a high level of accuracy. Looking forward, the optimization and real-world deployment of these models hold the potential to revolutionize cybersecurity workflows, reducing response times and improving threat detection capabilities in dynamic, high-risk environments.

APPENDIX A

A.1 Project Schedule

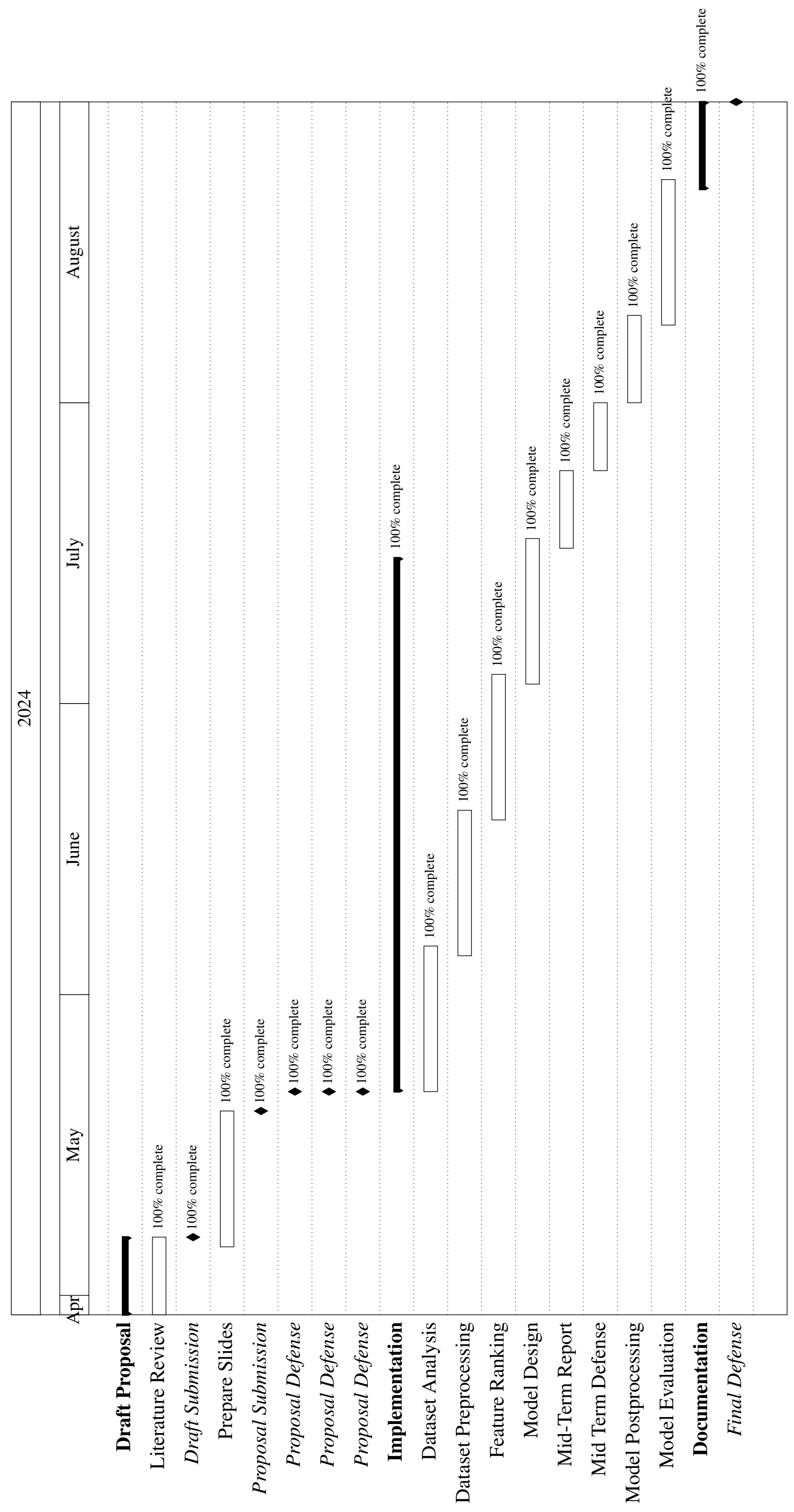


Figure A.1: Gantt Chart showing Expected Project Timeline.

A.2 Literature Review of Base Paper- I

Author(s)/Source: Raut, Madhu, et al.	
Title: Insider Threat Detection using Deep Learning: A Review	
Website: `https://ieeexplore.ieee.org/document/9315932`	
Publication Date: 18 January 2021	**Access Date: May, 2024**
Publisher or Journal: IEEE	**Place:** Thoothukudi, India
Volume: n/a	**Issue Number:** n/a
Author's position/theoretical position: student at Defence Institute of Advanced Technology	
Keywords: insider threat detection, deep learning, log analysis, anomaly detection, ELK stack	
Important points, notes, quotations	**Page No.**
1. Research studies a series of occurrences or trends in order to identify aberrant behaviour.	**2**
2. Making a cyberweapon is known as weaponization. By way of social engineering, for instance	**1**
3. UEBAis coupled with Security Identity Event Management (SIEM) to combat this sensitive domain.	**2**
4. LSTM is a special category of RNNs with the capability of learning long-term temporal dependencies	**3**
5. CERT is a very comprehensive public dataset and is diverse in its content	**4**
Essential Background Information: Advanced persistent threats (APTs) and disgruntled employees are only two examples of the many varieties and motivations of insider threats. According to the 2018 Insider Threat Report, privileged account information is most susceptible to insider assaults. Of all insider threats, there are 47% malevolent/intentional insiders and 51% accidental/unintentional insiders.	
Overall argument or hypothesis: UEBA design: scoring (on-going monitoring); ingestion (from many data sources); correlation (distillation and integration); analysis (using deep learning and machine learning); Take action (against entities and users).	
Conclusion: Because RNN-based techniques can handle temporal log data, they were deemed to be the best option. CNN proven to be yet another effective approach when paired with RNN.	
Supporting Reasons	
1. It makes use of Kafka. After going through a Kafka topic, the combined logs from Logstash are given to the best Deep Learning model.	**2.** New streams of log data are used to continuously train and test this model.
3. Anomaly Scores are the predictive outcomes of the Deep Learning model.	**4.** After being routed through a Kafka topic, these outcomes are indexed into Elasticsearch for additional examination.
5. The predictions made by the Deep Learning model, along with the current observed values and anomaly scores, are indexed into Elasticsearch.	**6.** This stored data is visualized as graphs on the Kibana dashboard in real-time.
Strengths of the line of reasoning and supporting evidence: For time-series evaluation of logs, the RNN (LSTM) architecture is used to learn the behavioral pattern of logs and calculate their anomaly scores.	
Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: covered the importance of insider threat detection as well as the combination of deep learning and log-based anomaly detection, and we've tried to arm the readers with the necessary knowledge to embark on the work of locating a needle in a haystack that is insider threat detection. The creation and gathering of insider threat data may occupy a whole scientific domain. In this context, few-shot learning-based techniques can be applied. While there are fully or partially developed answers for some of these problems, the remaining ones are still being investigated.	

A.3 Literature Review of Base Paper- II

Author(s)/Source: Sharma, Balaram, et al.	
Title: User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder – Insider Threat Detection	
Website: `https://dl.acm.org/doi/10.1145/3406601.3406610`	
Publication Date: 03 July, 2020	**Access Date: April.2024**
Publisher or Journal: association for computing machinery (ACM)	**Place:** Nepal
Volume: n/a	**Issue Number:** n/a
Author's position/theoretical position: Engineer at LogPoint	
Keywords: Security and privacy, Intrusion/anomaly detection and malware mitigation, Intrusion detection systems	

Important points, notes, quotations — **Page No.**

1. Research studies a series of occurrences or trends in order to identify aberrant behaviour. **2**
2. The CERT dataset spans 502 days and includes users' daily actions within an organisation. The activities of the 1000 distinct users in the sample are spread out over 502 days. **3**
3. Unattended The model is prepared using LSTM RNN. Because the LSTM RNN model can comprehend sequential dependencies, it has been frequently employed for temporal data analysis. **6**
4. The threshold value selection determined the TPR, FPR, and Accuracy performance characteristics. **7**

Essential Background Information: Insider threats come in a wide variety of forms, with the two most common types being dissatisfied employees and advanced persistent threats (APTs). The 2018 Insider Threat Report states that insider attacks are most likely to target privileged account information. There are 51% accidental/unintentional insiders and 47% malevolent/intentional insiders out of all insider threats.

Overall argument or hypothesis: The proposed approach uses an LSTM-based Autoencoder to model user behaviour based on session activities, which enables it to identify anomalous data points.

Supporting Reasons

1. As per Gartner UBA is defined as a cybersecurity process about the detection of insider threats, targeted attacks, and financial fraud.	**2.** The user activity is simulated and evaluated using the CERT insider threat dataset. The dataset contains raw activity data in five distinct event categories. overall statistics of the data i. Device events ii. Visited HTTP URLs iii. Activity log-in and log-off iv. Details about emails versus file access.
3. Within 502 days, 1000 users generate 32,770,227 events (log lines).	**4.** The input sequence is given to the network during the training phase, where the encoder encrypts the data. as the decoder works to piece together the original data.

Strengths of the line of reasoning and supporting evidence: The experiment produced an Accuracy of 90.17%, while the TPR and FPR were 91.03%, and 9.84% respectively.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence:
A large amount of work was put into feature engineering in this study project. Moreover, during feature extraction, it is always possible to overlook some important information. Thus, in the future, the features required for behavioural pattern profiling can be constructed using deep learning algorithms.

A.4 Literature Review of Base Paper- III

Author(s)/Source: Abir Rahali and Moulay A. Akhloufi	
Title: MalBERTv2: Code Aware BERT-Based Model for Malware Identification	
Website: `https://www.mdpi.com/2504-2289/7/2/60`	
Publication Date: 24 march 2023	**Access Date:** march,2024
Publisher or Journal: MDPI - Publisher of Open Access Journals	**Place:** n/a
Volume: n/a	**Issue Number:** n/a
Author’s position/theoretical position: Researcher	
Keywords: malware detection; natural language processing; transformer-based model	
Important points, notes, quotations	**Page No.**
1. NLP’s ultimate objective is to find the keyword, read the content, and understand any pertinent context.	**1**
2. In AI-based domains, language modelling enhanced the transfer learning tasks.	**1**
3. BERT is a creative, extensively applied solution that is utilised in both business and academia.	**2**
4. The size and calibre of the training datasets have a major impact on how well the pre-trained models perform.	**16**
5. The models are evaluated using the following performance metrics: accuracy, Matthews correlation coefficient (MCC), precision (mc), recall (mc), F1- score (mc), and AUC are common classification problem metrics.	**22**
Essential Background Information: The use of pre-trained models on larger datasets to fine-tune specialised datasets for certain domains has become possible thanks to transfer learning-based techniques.	
Overall argument or hypothesis: Recent developments in natural language processing (NLP) models can help cybersecurity operations proactively detect different types of threats. Using a pre-trained language model named MalBERTv2, they provide a novel method in this study for capturing the relevance and significance of the Malware/Goodware (MG) datasets.	
Conclusion: The MalBERTv2 classifier features a fully connected layer for the prediction probabilities in addition to a BERT layer block with the same weights as the pre-trained BERT. We present MalBERTv2, an end-to-end malware/goodware language paradigm for malware categorisation, by combining all these components.	
Supporting Reasons	
1.A series of studies have been conducted to tackle cybersecurity threats using machine learning (ML) and deep learning (DL) tools.	**2.**With the exponential growth of NLP applications, DL algorithms, and specifically transformer-based (TB) architectures, have gained popularity in solving complex problems.
3.They propose a pre-tokenization process to present the features..	**4.**They propose MalBERTv2, an improved version of the MalBERT approach for malware detection representation by creating a full pipeline for a code-aware pre-trained language model for MG detection.
Strengths of the line of reasoning and supporting evidence: As a layer in the model pipeline, they employ a classifier that makes use of bidirectional encoder representations from transformers (BERT). Our model’s effectiveness is assessed across various datasets, yielding a weighted f1 score that falls between 82% and 99%.	
Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: They were unable to properly compare the model with current techniques due to the limitations, which included the absence of benchmarks for malware/goodware detection. They plan to expand the research in the future by decreasing the feature generation and prediction process and enhancing the platform execution time in its entirety.	

A.5 Literature Review of Base Paper- IV

Author(s)/Source: Pérez-Gomariz, Mario , et al.	
Title: Lm-Hunter: An Nlp-Powered Graph Method for Detecting Adversary Lateral Movements in Apt Cyber-Attacks at Scale	
Website: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4807938`	
Publication Date: October,2023	**Access Date:** March, 2024
Publisher or Journal: Elsevier, SSRN	**Place:** n/a
Volume: Proceedings of the First Workshop on Natural Language Interfaces	**Issue Number:** n/a
Author's position/theoretical position: Researcher	
Keywords: Anomaly Detection, Advanced Persistent Threat, Lateral Movements, Cybersecurity Knowledge Graphs, Natural Language Processing, Transformer	
Important points, notes, quotations **Page No.** 1. merged all log files into a unique event log dataset with 4 million events from 100 servers 10k users. **11** 2. seq2seq models are efficient solution for detecting lateral movements on large enterprise networks. **18** 3. LMHunter covers a larger threat landscape by considering all types of login activity in the network (RDP, remote PowerShell, WMI, SMB, etc.) **19** 4. The model learns the roles of the machines based solely on the user login activity in the network. **19**	
Essential Background Information: With an average total cost of a data breach reaching USD 4.45 million – USD 9.48 million in the US – companies continue increasing their cybersecurity budgets to minimize the time to identify and contain breaches (IBM, 2023)	
Overall argument or hypothesis: UEBA (User and Entity Behavior Analytics) or graph analysis have been proposed to identify APT actor activities.	
Conclusion: TLM-Hunter, a new method that leverages graphs and NLP technologies for hunting APT adversaries at scale. LM-Hunter uses seq2seq models to process user login activity sub-graphs to identify the most suspicious lateral movements in the network.	
Supporting Reasons	
1. We tested our system in a real-world cybersecurity incident at a Fortune 500 company.	**2.** LM-Hunter identified numerous lateral movement activities of the threat actor in a network with thousands of machines and users.
3. The fully unsupervised methodology, coupled with the lightness of the models, makes our method suitable for hunting threats in real-world scenarios.	**4.** They facilitate the application of our method by releasing LM-Hunter as an open-source tool for Threat Hunting and Incident Response teams.
Strengths of the line of reasoning and supporting evidence: In contrast to UEBA solutions, LM-Hunter provides a holistic view of the user's lateral movements in the network. The use of graphs facilitates the contextualization of user activities, enhancing the ability of cybersecurity teams to identify stealth APT actors moving through the network.	
Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: After comparing the results of 10 executions, the outputs of the models showed minimal fluctuation. We got the best results when training the models between 65% and 85% accuracy rates. For further future research, performance can be improved.	

A.6 Literature Review of Base Paper- V

Author(s)/Source: AHEIDING, FREDRIK, et al.	
Title: Devising and Detecting Phishing Emails Using Large Language Models	
Website: `https://ieeexplore.ieee.org/document/10466545`	
Publication Date: 11 March 2024	**Access Date:** Feb, 2024
Publisher or Journal: IEEE	**Place:** Coimbatore, India
Volume: 12	**Issue Number:** n/a
Author's position/theoretical position: Research Student	
Keywords: Phishing, large language models, social engineering, artificial intelligence.	

Important points, notes, quotations — **Page No.**

1. The work looks into a sequence of events or patterns for detecting abnormal behavior. **2**
2. LLMs were used to detect phishing emails. **5**
3. To create emails using LLMs, ChatGPT was used. **6**
4. ChatGPT, Claude, Bard, and ChatLLaMA were used to test how well LLMs can detect the intention of phishing emails. **7**

Essential Background Information: Because LLMs have a talent for mimicking human language and thought processes, they are ideal for creating phishing emails. Similar to LLMs, phishing attempts to leverage a few pieces of information about the victim to produce content that seems pertinent and realistic. Because their emails are automatically created, LLMs shorten the time needed to launch general-target phishing attacks while also increasing the success rate because their emails are of a high calibre.

Overall argument or hypothesis: How LLMs were used to develop and send phishing emails, as well as how LLMs were used to identify phishing emails. Big language models also demonstrate promise for improving cybersecurity education by tailoring it to each user's unique requirements.

Conclusion: Additionally, the language models shown encouraging skills to identify phishing emails' intent, occasionally identifying malicious intent in less evident phishing emails and surpassing humans in this regard. The four tested models (GPT-4, Claude, Bard, and LLaMA) varied greatly in terms of performance and stability; Claude produced the most reliable and practical outcomes.

Supporting Reasons

1.According to their research, there is no one-size-fits-all method for developing phishing emails or assisting consumers in avoiding them..

2.What deters one individual from clicking on phishing emails may fool another. As a result, the strategies need to be unique.

3.This personalisation, which may be utilised maliciously or preventively, is something that LLM are quite skilled at creating.

4.The effectiveness of LLMs in identifying the purpose of phishing emails was tested using ChatGPT,Claude,Bard, and ChatLLaMA.

Strengths of the line of reasoning and supporting evidence: Additionally, Claude offered helpful advice on how to respond to phishing emails, like visiting the business's official website to confirm a possible gift card offer..

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The capabilities of LLMs are being researched at a rapid pace, and findings become dated soon. Rather than serving as a conclusion, the trials presented in this article should be viewed as a starting point for more research. Their research objective for the future is towards putting LLM-based, customised phishing training into practice.

REFERENCES

[1] Mohammed Nasser Al-Mhiqani, Rabiah Ahmad, Z. Zainal Abidin, Warusia Yassin, Aslinda Hassan, Karrar Hameed Abdulkareem, Nabeel Salih Ali, and Zahri Yunos. A review of insider threat detection: Classification, machine learning techniques, datasets, open challenges, and recommendations. *Applied Sciences*, 10(15), 2020.

[2] Fredrik Heiding, Bruce Schneier, Arun Vishwanath, Jeremy Bernstein, and Peter S. Park. Devising and detecting phishing: Large language models vs. smaller human models, 2023.

[3] Dahye Kim, Dongju Park, Honghyun Cho, and Kang. Insider threat detection based on user behavior modeling and anomaly detection algorithms. *Applied Sciences*, 9:4018, 09 2019.

[4] Abir Rahali and Moulay A. Akhloufi. Malbertv2: Code aware bert-based model for malware identification. *Big Data and Cognitive Computing*, 7(2), 2023.

[5] Madhu Raut, Sunita Dhavale, Amarjit Singh, and Atul Mehra. Insider threat detection using deep learning: A review. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 856–863, 2020.

[6] Balaram Sharma, Prabhat Pokharel, and Basanta Joshi. User behavior analytics for anomaly detection using lstm autoencoder - insider threat detection. In *Proceedings of the 11th International Conference on Advances in Information Technology*, IAIT ’20, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *CoRR*, abs/1710.00811, 2017.