Wrocław University of Science and Technology
Faculty of Information and Communication Technology

Field of study: **Cybersecurity (CBE)**
Speciality: **Communication network security (CBS)**

BACHELOR THESIS

Cybersecurity Log Analysis With Artificial Intelligence

Tomasz Przybyszewski

Supervisor
dr inż. Michał Kędziora

Keywords: Artificial Intelligence, Log Analysis, Log Summarization, Anomaly Detection

WROCŁAW 2026

To my beloved parents,

for their endless support, love, and faith in me.

ABSTRACT

This thesis explores the application of transformer-based Large Language Models, specifically the fine-tuned Mistral 7B model, in cybersecurity log analysis. The research focuses on improving log summarization and anomaly detection by leveraging AI techniques to process unstructured log data from diverse systems. A web application was developed, enabling real-time analysis and user interaction with the fine-tuned model. Evaluation metrics such as ROUGE and BERTScore demonstrated improvements in the model's performance compared to non-fine-tuned version, particularly in providing actionable insights for security operations. The study also highlights challenges, such as ensuring data privacy and scalability, and offers recommendations for future research, including expanding datasets and exploring secure deployment options. The findings confirm the potential of LLMs to enhance cybersecurity practices by automating log analysis.

STRESZCZENIE

Praca inżynierska bada zastosowanie dużych modeli językowych opartych na architekturze transformer, w szczególności dostrojonego modelu Mistral 7B w analizie logów cyberbezpieczeństwa. Badania koncentrują się na poprawie podsumowywania fragmentów logów i wykrywania anomalii poprzez wykorzystanie technik sztucznej inteligencji do przetwarzania nieustrukturyzowanych logów pochodzących z różnych systemów. Opracowano aplikację internetową umożliwiającą analizę logów w czasie rzeczywistym i interakcję użytkownika z modelem. Metryki takie jak ROUGE i BERTScore, wykazały poprawę wydajności modelu w porównaniu z wersją niedostrojoną, szczególnie w zakresie dostarczania przydatnych informacji w kontekście cyberbezpieczeństwa. W pracy podkreślono również wyzwania, takie jak zapewnienie prywatności danych i skalowalności, a także przedstawiono zalecenia dotyczące przyszłych badań, w tym powiększenia zbiorów danych i zbadania bezpiecznych opcji implementacji systemu. Wyniki badania potwierdzają potencjał dużych modeli językowych w zakresie poprawy praktyk cyberbezpieczeństwa poprzez automatyzację analizy logów.

CONTENTS

LIST OF FIGURES

LIST OF LISTINGS

LIST OF TABLES

1. INTRODUCTION

In the current time of technological advancements, and the rapid increase of data quantity and variety has brought new challenges and opportunities in the realm of cybersecurity. Cybersecurity Ventures forecasts that by 2025, global cybercrime damages will reach $10.5 trillion annually, marking a substantial increase from previous years [16]. In 2022, enterprises experienced a 35% year-over-year increase in logging data, resulting in billions of entries requiring real-time analysis [18].

The most critical data are logs, files generated by the systems and applications that are detailed records of system activities, errors, and interactions with other system components. Log files are essential for monitoring system state, identifying troubleshooting issues, and most importantly, detecting and mitigating security threats. And as frequency of the attacks increase, current analysis approaches are becoming less effective in addressing the problem.

Artificial Intelligence has proven to be an impactful technology for cybersecurity solutions. It can process vast amounts of data, identify patterns, and provide insights. A significant advancement in the field of Artificial Intelligence is the development of transformer-based models, such as GPT models from OpenAI, which specialize in handling text based data, therefore making them them ideal solution for log analysis given the text based structure of log files.

However, integrating artificial intelligence into log analysis presents new challenges. Difficulties in handling unstructured log data, ensuring the precision of AI model outputs, and maintaining data privacy present issues that must be addressed to fully leverage AI in cybersecurity contexts.

This work focuses on the application of Large Language Models in the analysis of cybersecurity logs, it focuses on how transformer-based models can improve tasks like summarization and anomaly detection.

1.1. PURPOSE OF THE WORK

The purpose of this thesis is to demonstrate cybersecurity log analysis with artificial intelligence by creating an application that implements transformer-based Large Language Model that helps with the analysis of cybersecurity logs. By investigating how these advanced models can enhance tasks such as log summarization and anomaly detection, the work aims to demonstrate the potential benefits of integrating artificial intelligence into cybersecurity practices. The thesis seeks to address the challenges associated with handling vast amounts of unstructured log data, ensuring precision in model outputs, and maintaining data privacy.

1.2. SCOPE OF THE WORK

The scope of this thesis involves the design, implementation, and evaluation of a system that utilizes transformer-based Large Language Model for analyzing cybersecurity logs. It covers a review of existing literature on log analysis and the use of artificial intelligence in the cybersecurity field, selection and fine-tuning an appropriate AI model to process and summarize log data, and developing a web application that allows the user to interact with the fine-tuned model that can provide real-time log analysis. The evaluation of the model's performance is calculated using metrics such as ROUGE and BERTScore, log sample case studies are also performed, with comparisons made against standard non fine tuned model for improvement evaluation. The study is based on log files from sixteen different systems including Linux, Windows, and Apache to ensure the model's applicability across different environments.

1.3. STRUCTURE OF THE WORK

The structure of the thesis is organized as follows:
Chapter 1: Introduction - This chapter describes the motivation, purpose, scope, and structure of the work. It emphasizes the critical role of logs in cybersecurity and outlines the challenges and opportunities in leveraging AI for log analysis.
Chapter 2: Theoretical Background - Discusses foundational concepts of the cybersecurity logs, traditional log analysis methods, and the role of artificial intelligence within the cybersecurity field.
Chapter 3: Related Work - A review of existing literature and tools related to the application of LLMs in cybersecurity, with a focus on their limitations and how this work aims to address these constraints and bridge the identified gaps.
Chapter 4: Methodology - Details the research design, including system architecture, data sourcing, fine-tuning strategies, and evaluation methodologies.
Chapter 5: Implementation - Discusses the technical implementation of the system, covering frontend and backend development, database integration, and the processes involved in model fine-tuning.
Chapter 6: Analysis of Model Effectiveness - Presents a detailed evaluation of the fine-tuned model's performance using various metrics and case studies based on particural log samples.
Chapter 7: Conclusions and future work - Summarizes the research findings, highlights the contributions of this work, identifies challenges and potential areas for future exploration.

2. THEORETICAL BACKGROUND

Artificial Intelligence has revolutionized numerous fields by providing systems the ability to learn and adapt through experience instead of instead of relying on rigidly predefined instructions. One of the most impactful advancements in AI is the development of Transformers, a model architecture that powers state-of-the-art NLP systems like ChatGPT allowing them to work with unstructured text data such as log files, which are present in every modern computing environment. By leveraging Large Language Models, it is now possible to automate tasks like log summarization and log analysis, reducing the burden on human analysts and improving response times on cybersecurity incicents.

2.1. LOGS IN CYBERSECURITY

Log files are a special type of files as all activities that have taken place in the monitored system are written in them. This applies to each task and service. A record in such a file usually consists of: service status, timestamp and information specific to the documented service. The main application of this type of file is it's use in analyzing the system behavior in the past for the purposes of troubleshooting, monitoring and security analysis. Log files serve as a valuable resource for system and security administrators to understand system behavior and detect anomalies [11].

Log files can be generated from different sources such as operating systems, applications, or even network appliance. These files can also serve different purposes. System logs for example provide insight into operating system activities, such as hardware errors and process execution. Application logs offer insights into how software interacts with the rest of the system that it is installed on, or any other external services within it, these also enable troubleshooting of such application. Security logs, on the other hand, play a crucial role in identifying potential breaches by tracking events like unauthorized access attempts or any suspicious network traffic. By integrating and analyzing these logs that may originate from various sources, security analyst can obtain insight into the state of the system [3].

In the event of a cyber attack, log files are used in postmortem intrusion detection analysis. From such analysis it is possible to find out how the attack was carried out. Efficient finding of the appropriate record in such a file speeds up the process of obtaining evidence for the exploit. It also allows the system to be secured in the future because based on information from log files rules can be created in Intrusion Detection And Intrusion Protection systems to protect against the possibility of the exploit being used again [7].

In small computer systems, log files can be customized to the specific format that would be required for further processing, the problem occurs when log files come in huge quantities from a large number of different devices, in which case the number and variety of formats of these files significantly slows down the analysis process [11].

2.2. TRADITIONAL METHODS FOR LOG ANALYSIS

Traditional methods for cybersecurity log analysis primarily encompass signature-based and statistical or rule-based approaches. While these techniques have been instrumental in identifying known threats, they exhibit certain limitations, particularly against novel or sophisticated attacks.

Signature-based detection is based on comparing log data that was generated by potentially malicious baehaviours within the monitored system, against a repository of known threat signatures, where each signature within the database is representing unique value associated with specific malicious activities. Tools that utilize this method scan logs to identify these patterns, when pattern match is found such system is generating an alert. This approach is effective for detecting known threats but is less capable of identifying new or evolving attacks that are lacking existing signatures [9].

Statistical methods establish baselines of normal system behavior by analyzing historical log data generated by such system. When system behaviour is changed in any way, alert is generated that informs about potential anomaly that could indicate malicious behaviour within the monitored system. Rule-based systems utilize predefined rules, often crafted by cybersecurity experts, to identify suspicious activities based on known indicators of compromise and patterns specific to malicious behavior. While these approaches can detect a range of issues, their effectiveness heavily depends on the quality of the predefined rules and statistical models [10].

Despite their utility, traditional log analysis methods face several challenges. Detection of unknown threats is a significant issue. Signature-based systems are ineffective against novel attacks without existing signatures, making systems vulnerable to new threats. These systems also require regular updates of the databases that contain signatures and rule sets to keep pace with emerging threats. These limitations generate the need for more adaptive and intelligent log analysis techniques, such as those leveraging artificial intelligence and machine learning, to enhance detection capabilities and reduce reliance on manual updates.

2.3. ARTIFICIAL INTELLIGENCE AND NATURAL LANGUAGE PROCESSING

The term Artificial Intelligence (AI) applies to the computer simulation of human intelligence. It is designed to perform tasks typically requiring human intelligence, such as decision-making, learning, and problem-solving. AI covers a broad range of subfields, such as Machine Learning (ML), robotics, and expert systems. Primary goal of AI is to do tasks usually human is required to do such as: knowledge presentation, natural language processing (NLP) or even autonomous driving.

Natural language processing (NLP) is a sub field of Artificial Intelligence, that focuses on application of computational algorithms in a way that allows computers to communicate with humans. The main goal of NLP is to enable computers to learn and communicate knowledge in a way that allows reader to understand the intended meaning of the communicate. All of this is achieved through various computational models and algorithms. Core component required to complete NLP tasks is language model.

Language models are complex algorithms that calculate a probability distribution describing the likelihood of a string. It can predict words that are likely to come next in a text and suggest completions of it. So they are the heart of a broad range of natural language tasks. Modern language models use deep learning architectures like transformers to be able to complete complex language tasks with high accuracy. [23].

The transformer architecture was one of the most important breakthroughs in deep learning, and even more so for natural language processing. It was introduced by the researchers at Google in their paper published in 2017 called "Attention Is All You Need". Unlike earlier models like recurrent neural networks, transformers based models process sequences all at once instead of one step at a time, which allows them to handle longer texts more efficiently. The key to their success is the attention mechanism, which helps the model focus on the most important tokens within a sequence, no matter where they appear [30]. That mechanism makes transformer architecture not only extremely powerful for finding patterns in large datasets but also at the heart of state-of-the-art NLP systems such as GPT and BERT. Though transformers are highly effective for general natural language processing tasks, their full potential can be unlocked through fine-tuning, which customizes the pretrained models for specific tasks.

Fine tuning is a process in which pretrained model is trained further on much smaller dataset of labelled data, in which its weights are updated and usually extra layer in neural network is added. Fine tuning allows AI model to be customised for a specific task that is required to be performed in a particular application such as text classification or sentiment analysis. [22]

2.4. OVERVIEW OF LARGE LANGUAGE MODELS

The development of large language models has evolved significantly since the introduction of the ELIZA [25] program recognised as the world's first chatbot. Revolutionary progress occurred with the introduction of transformer architecture, which has become foundational in natural language processing.

First natural language processing model after original transformer model that utilized Transformer architecture was GPT model proposed by OpenAI researchers in the article called "Improving Language Understanding by Generative Pre-Training" [21]. It addressed the limitations of supervised learning due to the need for labeled data in huge quantities. Researchers proposed a semi-supervised approach that combined unsupervised pre-training on 7000 unique books from variety of genres that allowed model to learn to condition on long-range informations and supervised fine-tuning on labeled data.

In the same year Google researchers published BERT [4] that stands for Bidirectional Encoder Representations from Transformers. It employed same two step approach to training as as GPT however the pre-training process was bidirectional. It processed text from left-to-right and right-to-left, allowing deeper understanding of language nuances. However for BERT still required fine tuning for specific applications.

In 2020 OpenAI introduced GPT-3 [2] that had 175 bilion parameters and marked substantial increase in models size compared to previous models. Also unlike traditional NLP systems it could perform tasks without extensive fine-tuning on large datasets. It could perform tasks with only few examples or even simple instructions and generate text that is often indistinguishable from the one written by humans.

In 2023, Meta introduced LLaMA [28] that focused on efficiency, and demonstration that it could achieve state-of-the-art performance only on public datasets and without the need for excessively large models. LLaMA-13B outperformed GPT-3 on most benchmarks despite being significantly smaller in size. The paper published by Meta also contributed to the conversation about the need for models to not only be pwerfull but also socially responsible.

In the same year Mistral AI created model Mistral 7B [8] that outperformed not only LLaMA 34B but also LLaMA 2 13B [29] models with even less parameters. Mistral 7B was also crafted for the ease of fine-tuning across the various tasks. The model was fine tuned to follow instructions resulting in Mistral 7B-instruct.

2.5. CHALLENGES IN USING LLMS FOR CYBERSECURITY LOG ANALYSIS

Integrating Large Language Models into cybersecurity log analysis presents challanges, regarding data privacy, security, and computational demands. To train LLM's significant amount of data is required, data used for training may contain sensitive information such as personal information or confidential organizational data. Processing this kind of information raises significant privacy concerns, especially when utilizing cloud-based services, as it can lead to potential data leakage and unauthorized access. To guarantee compliance to data protection statutes such as the General Data Protection Regulation, it is required to protect confidential information when it is used for LLM purposes. Large Language Models are also vulnerable to attacks, where hackers try to exploit model vulnerabilities to extract confidential data or manipulate models outputs in any way. To mitigate these risks, organizations can implement secure and encrypted data pipelines, conduct on-premises model fine-tuning, or even employ data anonymization techniques before processing the data [12].

Deploying LLMs for log analysis also demands substantial computational resources, including high memory capacity and processing power, leading to increased energy consumption and operational costs. This resource intensity poses challenges for real-time log analysis and scalability, particularly in resource-constrained environments. To address these issues, organizations can utilize optimized, smaller-scale models, such as DistilBERT, and apply techniques like model quantization and pruning to reduce computational demands. Leveraging hardware accelerators, including Graphics Processing Units and Tensor Processing Units, can also allow enchancement of processing efficiency [24].

3. RELATED WORK

Recent advancements in natural language processing have led to the development of systems that enhance cybersecurity through log analysis, anomaly detection, and policy generation. This section reviews notable works, such as CYGENT, SecBot, and LogGPT, highlighting their contributions and limitations, providing context for the fine-tuning of Mistral 7B presented in this thesis.

CYGENT is a cybersecurity conversational agent with a log summarization function. It utilizes a fine-tuned GPT-3.5 Turbo model and achieves a BERTScore of over 97%. It simplifies log analysis tasks performed by system administrators and security analysts. The conversational agent was designed to provide users with relevant cybersecurity information, detect specific events within the logs, and summarize log files. The authors compared various GPT-3 models and Code-T5 models, concluding that the Davinci (GPT-3) model consistently outperformed the others in log analysis tasks [1].

SecBot is a chatbot designed to assist small and medium-sized enterprises in cybersecurity planning and management. It can recognize potential cyberattacks based on symptoms described by users and helps businesses make informed decisions on cybersecurity investments. In contrast to CYGENT Rasa was chosen as the framework for developing the conversational capabilities of SecBot, allowing it to interact efficiently with users and respond to their cybersecurity related queries [6].

LogGPT utilizes the GPT-3.5 Turbo model for log-based anomaly detection. The detection process involves three key components: first, the logs are preprocessed using the Drain log parser; then, a custom prompt is created to guide ChatGPT in anomaly detection. After that, the response is parsed to extract and format ChatGPT's output into usable results. The authors emphasize that prompt creation plays a crucial role in the framework's effectiveness, it is also worth mentioning that autors noticed LogGPT's tendency to generate high ammount of false-positive detections [20].

In the article An Assessment of ChatGPT on Log Data, the authors investigate ChatGPT's capabilities in handling log data, including its ability to extract log templates, summarize errors, detect anomalies, and predict future log events. The study reveals that while ChatGPT's performance in anomaly detection is limited and inconsistent, however it is great solution for AI based log parsing [17].

The study Harnessing GPT-4 for the Generation of Cybersecurity GRC Policies: A Focus on Ransomware Attack Mitigation investigates the potential of Generative Pre-trained Transformers (GPTs) in generating Governance, Risk, and Compliance (GRC) policies. The research assesses how effectively GPTs generate GRC policies by comparing them to those created by established security vendors and cybersecurity agencies. The results suggest that, in certain scenarios, GPT-generated policies may outperform traditional human-generated policies [14].

This thesis builds on works mentioned above by fine-tuning Mistral 7B to enhance both log summarization and anomaly detection for cybersecurity applications. Unlike prior studies, the fine-tuning process focuses on diverse log sources and uses a structured output format to improve clarity of generated responses. Additionally, low processing power requrements of Mistral 7B make this approach accessible for a wide range of users, from small to large enterprises as well as private users. Comparison of systems mentioned above can be seen in table 3.1.

Table 3.1: Comparison of Related Works

Feature	**CYGENT**	**SecBot**	**LogGPT**	**ChatGPT for Logs**	**This Work**
Focus	Log summaries	Cybersecurity	Anomaly detection	Parsing logs	Summarization & detection
Model	GPT-3.5 Turbo	Rasa	GPT-3.5 Turbo	ChatGPT	Mistral 7B
Strengths	Accurate outputs	SME assistance	Custom prompts	General parsing	Domain-specific, efficient
Limitations	No detection	Limited scale	False positives	Inconsistent results	Dataset-dependent
Contributions	Summarization	Broader use	Improved tuning	Structured results	Low resource requirements

4. METHODOLOGY

The user-facing component of the system was built with the Django framework. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design [5]. Given my familiarity with its architecture, I was able to efficiently build a system that is easy to use and easily scalable. The web application allows users to interact with Mistral conversational agent via an API. It is designed to focus on simplicity and convenience, while the API connection facilitates fast query processing, improving the speed and performance of the system.

4.1. SYSTEM ARCHITECTURE

Log Summarisation form 4.1 This module allows users to paste raw log data, that is then processed and sent via API to the fine tuned Mistral conversational agent. This module is designed to simplify the log handling process and improve the efficiency of data interpretation.

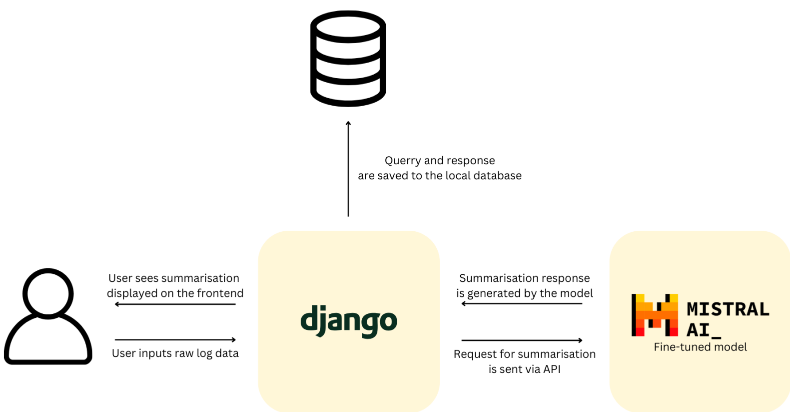


Fig. 4.1: Summarisation module architecture

Question Submission form 4.2 This module allows user to interact with conversational Mistral 7B agent. Once the user inputs their question, the system processes it and sends it to the Mistral agent via API connection. Responses generated by the model are presented to the user in the "Answer" Section of the form. It is important to point out that conversational model has access to the summarization generated by the fine tuned model, so user can ask questions about log fragment as well as general questions regarding cybersecurity terminology.

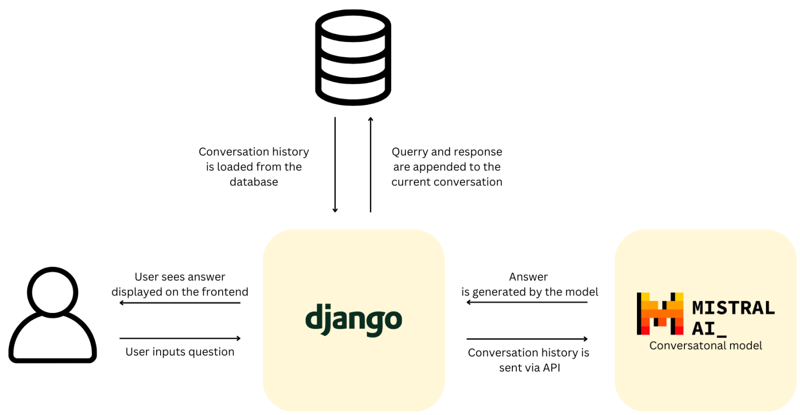


Fig. 4.2: Chatbot module architecture

Conversation History module 4.3 This module of the web application stores history of queries that are submitted by the user. Every question along with the corresponding answer, as well as the logs in their original and summarized form are stored in the SQLite database and shown in this module of application.

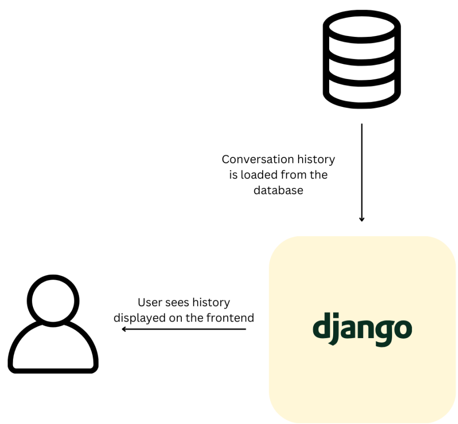


Fig. 4.3: History module architecture

4.2. CONVERSATIONAL AGENT

Mistral 7B was chosen for fine tuning based on its advantages cost efficiency as token based pricing for the the Mistral 7B when accessed via an API price per token offered a more economical solution compared to other NLP model providers. Also Mistral is optimized for resource efficiency to the point where it is possible to host it on personal grade computer. This is a critical advantage when it comes to data privacy. It allows the organizations and private users to deploy the system on premises, ensuring that data does not need to be transmitted to the third party cloud provider, leading to enhancement of data security.

4.3. LOG SOURCE AND PROCESSING

The dataset for model fine-tuning was created using logs sourced from a publicly available GitHub repository [32]. These logs originate from different systems including Windows, Linux, Android, OpenStack and Apache, making them a diverse source of data for building a dataset required for purpose of this thesis. As for dataset creation, the raw data was divided into segments ranging in length from 0 to 4096 tokens. Each log fragment of raw data was then processed individually. Logs were sent to Mistral Large model via API to create summarizations based on a predefined template. Summarizations generated by Mistral Large model were then reviewed, each log entry was validated in terms of descriptions of incidents shown in logs as well as fitting the predefined pattern.

4.4. FINE TUNING

Fine-tuning was utilized to enhance Mistral-7B's ability to process cybersecurity logs. This process allowed the model to learn unique patterns related to each system, improve response quality and description accuracy. The dataset consisted of two JSONL files, training file (138 samples) and test file (11 samples). Each JSONL entry had "user" message that contained raw log data and "assistant" message that provided summary based on log data from user message. Fine tuning process used Low-Rank Adaptation technique, that allowed updating model weights without retraining entire model. Training was carried out over 10 epochs with a learning rate of 1×10^{-4}, following recommendations from the Mistral documentation [15]. To direct model attention to important details custom prompt was crafted that instructed the model to break down generated information into "Key Log Activity", "Errors and Incidents" and "Security Recommendations" categories.

4.5. EVALUATION METRICS

Mistral-7B was evaluated by several metrics. The Recall-Oriented Understudy for Gisting Evaluation (ROGUE) [13] metric suite, that included ROGUE-1, ROGUE-2 and ROGUE-L scores for model evaluation. Where ROUGE-1 and ROUGE-2 quantify the overlap of single words as well as pairs of words that overlapped between prompt generated by the 7B model and prompt generated by Mistral Large model, validated manually. ROGUE-L measured the longest sequence that matched between two prompts. Along with ROGUE suite BERTScore [31] metric was utilized which uses embeddings from pre-trained BERT models to evaluate how closely the model's summaries match reference summaries at a contextual level.

5. IMPLEMENTATION

The system was implemented using the Python programming language, along with HTML and CSS, all of which were integrated within the Django framework. Django is a high-level Python web framework that facilitates rapid development and promotes a clean, maintainable design. Known for its speed, security, and scalability, Django served as the core of this system's development.

User interface was designed with Bootstrap framework. It helped to improve overall aesthetic appeal and user experience. Bootstrap is a powerful, feature-packed frontend toolkit that supports responsive design and provides a set of pre-designed CSS and JavaScript components that make it easier for developers to create modern, visually appealing, and functional user interfaces[27].

The system's backend utilized Django's built-in support for the SQLite Database. SQLite is a C-based library that implements a lightweight and efficient database engine. File format for SQLite is stable, cross-platform, and backward-compatible, making it well-suited for applications requiring a simple, dependable database solution [26]. Within the SQLite database user prompрrs as well as responses generated by Mistral model are stored.

The system also incorporated Mistral Language Model, developed by Mistral AI, which is open-weight model that can be universally customized and deployed to meet specific situational or individual requirements. Specifically, the Mistral 7B Instruct model was employed for handling query responses.

The project used the Mistral AI API accessed through the Mistral AI website to integrate the fine-tuned Mistral 7B model into the Django web application. This API allowed communication between the Django backend and the language model without the need for complex configurations or additional software. By directly connecting to the Mistral AI services, the application was able to process user inputs and generate responses in real time.

5.1. FRONTEND DEVELOPMENT

Frontend 5.1 is responsible for providing interface for submitting logs or questions and viewing results. User interface consists of two main sections that user can interact with as well ass two sections related to the application's usage history:

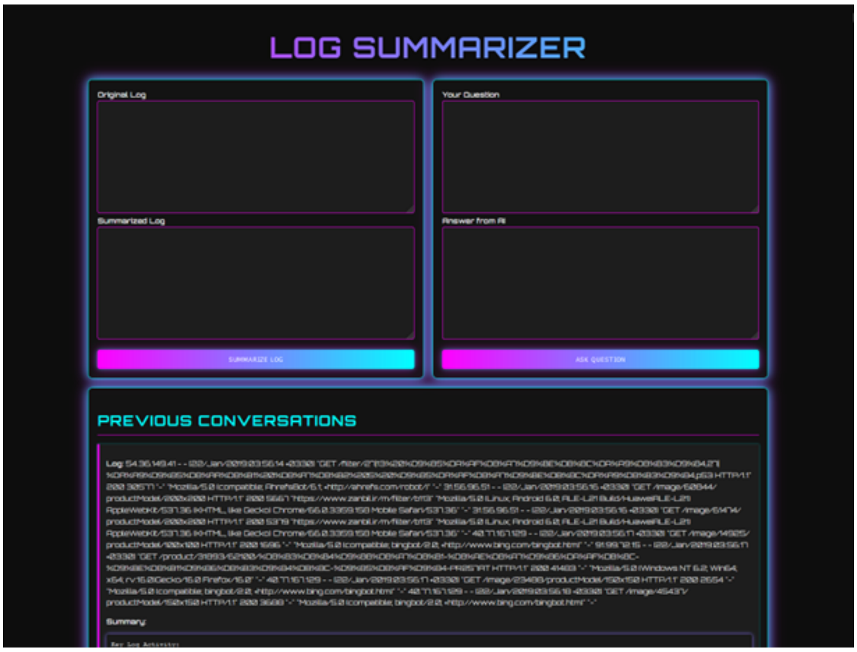


Fig. 5.1: Frontend

Log submission form 5.2 where user can submit their raw log data through a text area input, once log is submitted it is sent to the backend where it is processed and summarized using the fine-tuned Mistral model. Result is displayed under the submission form, and question submission forms.

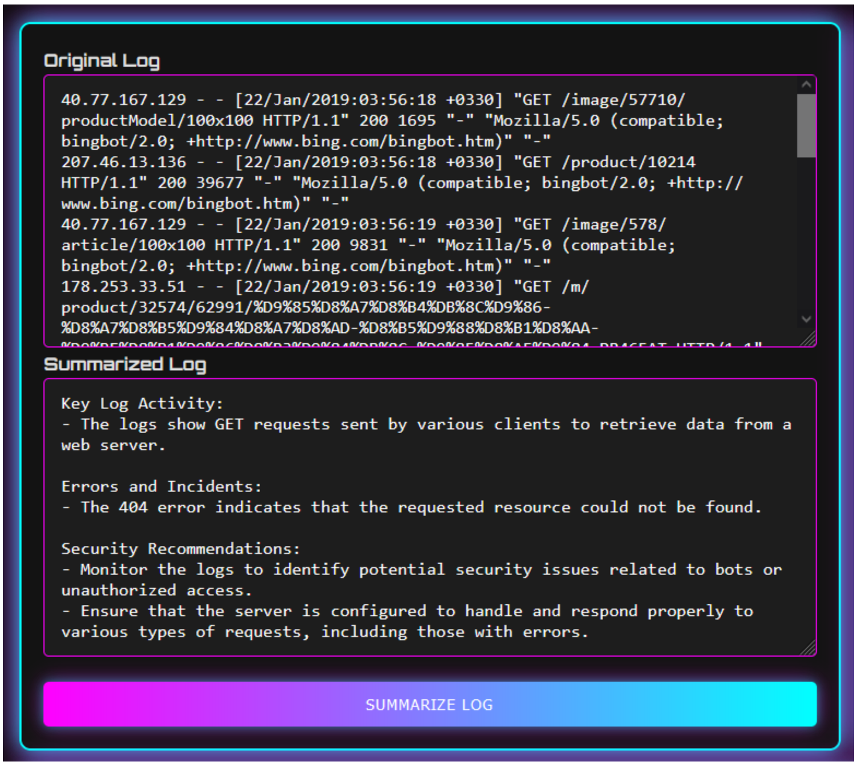


Fig. 5.2: Log submission form

Question submission form 5.3 user can submit question related or unrelated to the logs, once the question is submitted the whole conversation history as well as raw log data and summary is sent to the Mistral conversational model, based on all of that information answer is generated.

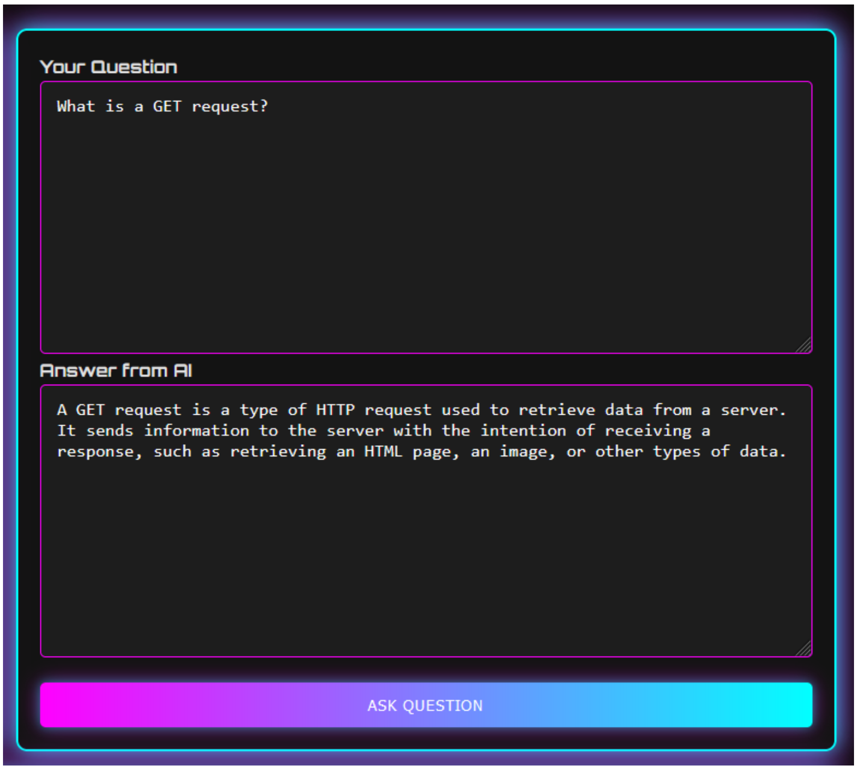


Fig. 5.3: Question submission form

Conversation history area 5.4 whole current conversation history is displayed in a designated area below both of the forms. The results area is structured to show original logs as well as summarised version of them. Questions asked by the user and answers generated by the model are also displayed in this area.

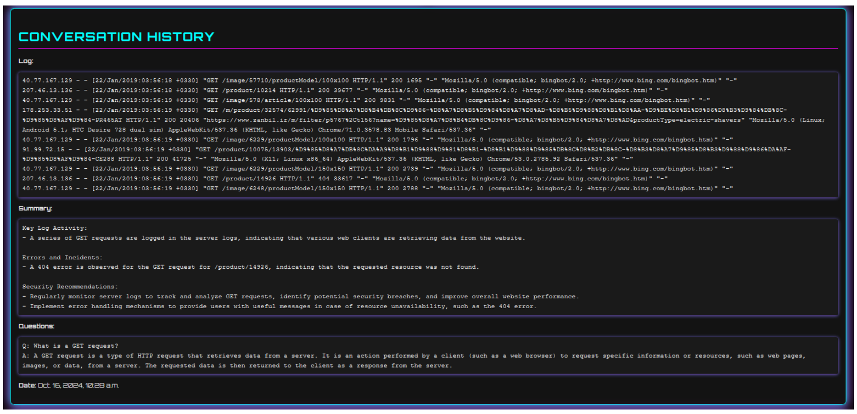


Fig. 5.4: Conversation history section

Previous Conversations 5.5 section is a list of previous conversations with AI model that user can view. This section provides summary of each session allowing user to revisit past interactions with the model.

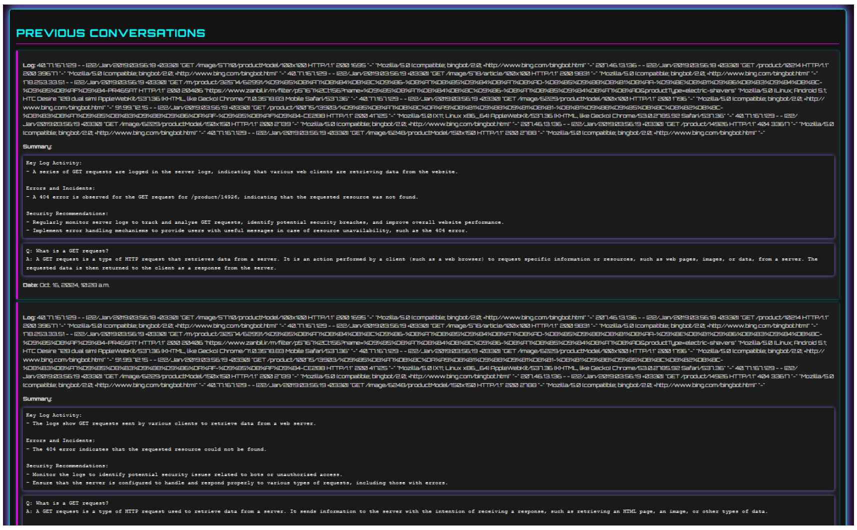


Fig. 5.5: Previous conversations section

5.2. BACKEND DEVELOPMENT

Backend of the web application integrates with API provided by Mistral AI to provide summarizations and question answering in real time. This interaction is handled by mistralai libary imported into the Django views.py file. The client is initialized with an API key, ensuring secure communication with the Mistral AI cloud. Backend handles two primary API interactions. One is for log summarisation and another one is for answering user questions. Function responsible for log summarizations, named "summarize_log" is responsible for connecting to conversational agent fine tuned specifically for log summarizations. Content of that function is shown in 5.1.

Function that is called when user asks question, is named "ask_question" and it connects to general purpose conversational agent, providing it with whole conversation history including summarisation provided by fine tuned agent. Structure of the function is in 5.2

```python
def summarize_log(log_text):
    try:
        response = client.agents.complete(
            agent_id="ag:3cf886ba:20241007:eng-agent:cdc6aee3",
            messages=[
                {
                    "role": "user",
                    "content": f"{log_text}:",
                },
            ],
        )
        summary = response.choices[0].message.content
        return summary
```

Listing 5.1: summarize_log function

```python
def ask_question(original_log, summarized_log, question, conversation_history):
    messages = []
    if conversation_history:
        history_entries = conversation_history.strip().split('\n')
        for entry in history_entries:
            if entry.startswith("Q: "):
                messages.append({"role": "user", "content": entry[3:].strip()})
            elif entry.startswith("A: "):
                messages.append({"role": "assistant", "content":
                ↪ entry[3:].strip()})

    messages.append({"role": "user", "content": f"{original_log}"})
    if summarized_log:
        messages.append({"role": "assistant", "content": f"{summarized_log}"})

    messages.append({"role": "user", "content": f"{question}"})


    response = client.agents.complete(
        agent_id="ag:3cf886ba:20241013:untitled-agent:35b91216",
        messages=messages,
    )

    answer = response.choices[0].message.content if response.choices else "No
    ↪ response."
    return answer
```

Listing 5.2: ask_question function

5.3. DATABASE INTEGRATION

The integration of a database into the Django application was essential for storing user inputs as well as summaries generated by the model. SQLite was chosen as the database backend for this project due to its simplicity, lightweight nature, and default integration with Django framework, making it ideal choice for for this project.

Django's Object-Relational Mapping (ORM) system was used for interaction with the database. It provides a high-level abstraction, that allows to work with database records using Python objects and classes instead of writing raw SQL queries. This abstraction not only simplifies database operations but also enhances security of an application as it is resilient to SQL injection attacks.

Main and only part of storage system is the *ConversationSession* 5.3 model defined in the *logs/models.py* file. This model represents the structure of the table within which data is stored for each user session. Each instance of *ConversationSession* corresponds to a unique conversation and includes the following fields:

— original_log – a TextField that sores raw log data submitted by the user

— summarized_log – a TextField that stores summarized version of logs generated by the AI model

— conversation_history – A TextField, that contains entire conversation between the user and the model, including summarized logs as well as question asked by the user.

— created_at – a DateTimeField, timestamp of when the conversation was started

```python
class ConversationSession(models.Model):
    original_log = models.TextField()
    summarized_log = models.TextField(blank=True)
    conversation_history = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Conversation started on {self.created_at}"
```

Listing 5.3: ConversationSession model

5.4. DATASET PREPARATION FOR FINE-TUNING

Fine-tuning the Mistral model required a dataset that reflects real-world cybersecurity logs. Data used for dataset creation was collected from the publicly available GitHub repository called Loghub [23]. This repository consists of wide variety of system logs, including Android, Apache, Blue Gene/L (BGL), Hadoop Distributed File System (HDFS), High-Performance Computing (HPC), Hadoop, HealthApp, Linux, Mac, OpenSSH, OpenStack, Proxifier, Spark, Thunderbird, Windows, and Zookeeper. These logs represent a wide range of systems, from mobile devices (Android) to large distributed computing frameworks (BGL). The diversity of logs was key to ensuring that the fine-tuned model could handle various types of queries and generalize across different system environments.

The first step of data preprocessing involved random sampling and segmentation. Given the diversity and size of the collected logs, random sampling method was emplyed to select representative log segments from each system category. Segments were selected with lengths ranging from 1 to 4096 tokens, corresponding to the Mistral model's maximum input sequence length. This token length was chosen to stay within the model's input processing limits, preventing truncation of log data during processing. Sample log segment is shown below.

```
Jun 30 20:53:06 combo klogind[19275]: Kerberos authentication failed
Jun 30 20:53:06 combo klogind[19277]: Authentication failed from
↪ 163.27.187.39 (163.27.187.39): Software caused connection abort
Jun 30 20:53:06 combo klogind[19277]: Kerberos authentication failed
Jun 30 20:53:06 combo klogind[19279]: Authentication failed from
↪ 163.27.187.39 (163.27.187.39): Software caused connection abort
Jun 30 20:53:06 combo klogind[19279]: Kerberos authentication failed
```

Listing 5.4: Log Sample

Once logs were segmented, the next step was to generate summaries based on each log fragment. For this purpose, the Mistral Large model was used via an API. The model was prompted to produce log summaries based on a predefined template designed to structure the output into three sections: "Key Log Activity", "Errors and Incidents", and "Security Recommendations" structured as shown in 5.5.

```
Key Log Activity:
- list of key log activity

Errors and Incidents:
- list of errors and incidents

Security Recommendations:
- list of security recommendations
```

Listing 5.5: Template structure

While the dataset creation process was heavily automated, it required manual intervention to ensure quality of generated output. Each case of inaccurate or insufficient model output was manually refined and corrected. Also python function for cleaning unnecessary sections generated by the model was utilized.

Final stage before actual fine-tuning was formatting dataset into JSON Lines (JSONL) format. It was chosen because of its compatibility with the Mistral model's API. The dataset, was initially stored in CSV format and contained two columns. First column contained raw log fragments, second column contained summarizations for each log fragment. Python script was used for conversion CSV file into JSONL file where each record was stored as a series of messages between a user (raw log data) and assistant (summary generated by the model). Example entry in JSONL format is shown in 5.6.

```
{"messages": [{"role": "user", "content": "03-17 16:14:21.466  2227  2227 V
↪  PhoneStatusBar: setLightsOn(true)\n"}, {"role": "assistant", "content":
↪  "Key Log Activity:\n- PhoneStatusBar status changed to lights
↪  on.\n\nErrors and Incidents:\n- None found.\n\nSecurity
↪  Recommendations:\n- Insufficient log data available to generate
↪  specific recommendations."}]}
```

Listing 5.6: JSONL entry

5.5. CHOSEN MODEL AND FINE-TUNING PROCESS

Fine-tuning of the Mistral 7B model was completed utilizing API provided by Mistral AI which allowed integration of a model without complex setup or infrastructure. Dataset for fine-tuning was uploaded in JSONL format directly to the platform without any reprocessing or normalization of the text. the model wast trained for 10 epochs and peak rate of 1e-4 as recommended for LoRA fine-tuning. After successful fine-tuning of the model conversational agent was created on the Mistral platform. Agent was provided with System prompt that contained the initial instructions for focusing the model capabilities on a defined task. Utilized prompt is shown in listing 5.7.

```
Template for Reporting Log Activity, Errors, and Security Recommendations

When provided with logs, please follow the structured format below. Ensure
↪  responses are clear, concise, and adhere to the requested sections.
Format

Key Log Activity:
- Provide a concise summary of notable activities found in the logs (e.g.,
↪  user login patterns, system alerts, or task completions). Include
↪  timestamps if relevant.

Errors and Incidents:
- List any errors and incidents separately where possible, defining errors
↪  as system malfunctions and incidents as potential security issues or
↪  anomalies.

Security Recommendations:
- Offer actionable recommendations for improving system security based on
↪  log activity. Specify high-priority items first.
```

Listing 5.7: System prompt

6. ANALYSIS OF MODEL EFFECTIVENESS

Evaluation of the models was conducted using several metrics: ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore. These metrics measure the quality and accuracy of the model's generated responses in comparison to reference data, where higher scores are representing stronger allignment with the expected response.

6.1. COMPARISON OF FINE TUNED MODEL TO STANDARD MODEL

Chart 6.1 shows average scores achieved by both models. On average, the fine-tuned model achieved scores that were 13.8% higher in the ROUGE-2 metric and 4.52% higher in the ROUGE-L metric compared to the standard model, while maintaining approximately the same performance in BERTScore. The only category where the fine-tuned model underperformed was in the ROUGE-1 metric, where it scored on average 5.7% lower than the standard model.

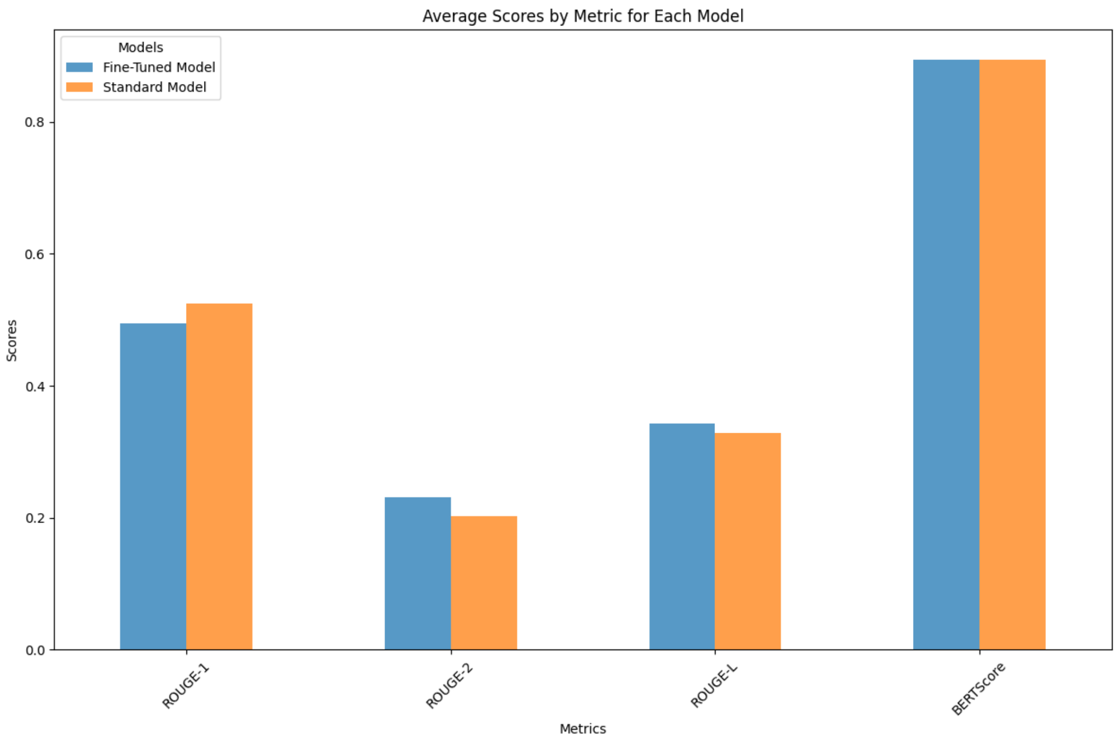
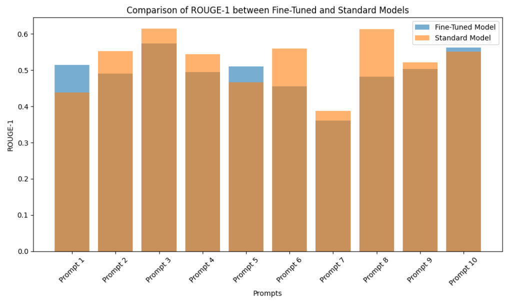
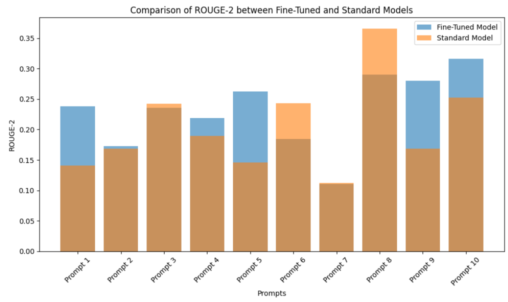


Fig. 6.1: Average Scores Comparison

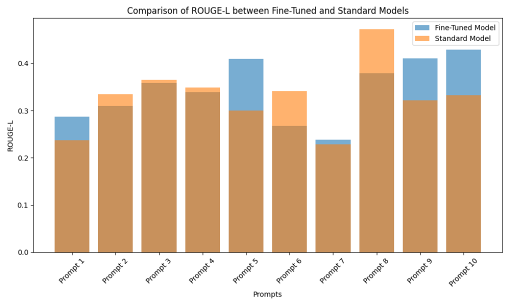
The calculation of the average scores was based on ten distinct prompts generated by the models. Scores achieved by each prompt can be seen in 6.2.



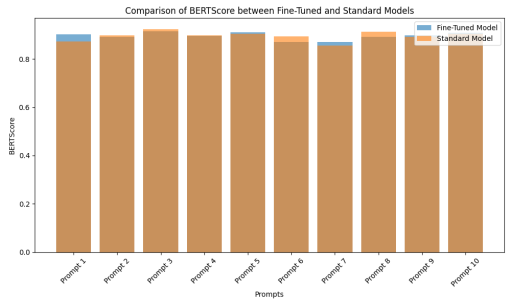
(a) ROUGE-1 Scores Comparison



(b) ROUGE-2 Scores Comparison



(c) ROUGE-L Scores Comparison



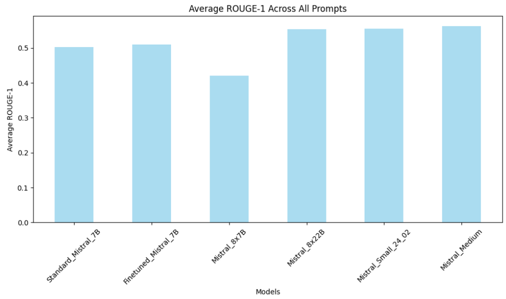
(d) BERTScore Comparison

Fig. 6.2: Model Score Comparisons by Metric

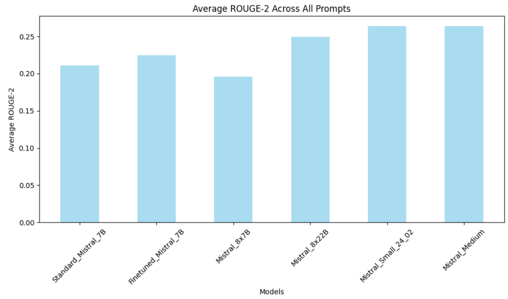
The improvement in ROUGE-2 and ROUGE-L scores that can be seen in responses generated by the fine-tuned model highlights its capability in identifying context within log data. The increased ROUGE-2 score indicates a better understanding of phrase-level relationships, that is crucial when summarizing log entries. In contrast, lower score in the ROUGE-1 metric, could be attributed to the fine-tuning process emphasizing contextual understanding over exact word matches. This trade-off suggests that the model prioritizes semantic precision, a valuable trait for anomaly detection and summarization tasks.

6.2. COMPARISON OF DIFFERENT MISTRAL MODELS

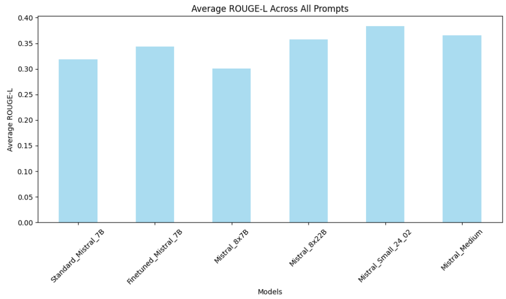
The fine-tuned model was also compared to other models provided by Mistral AI, comparison consisted of: Mistral 7B, Finetuned Mistral 7B, Mistral 8x7B, Mistral 8x22B, Mistral Small 24 02, and Mistral Medium. Other models were used with the zero-shot prompting technique, without any fine-tuning involved. Average scores can be seen in 6.3.



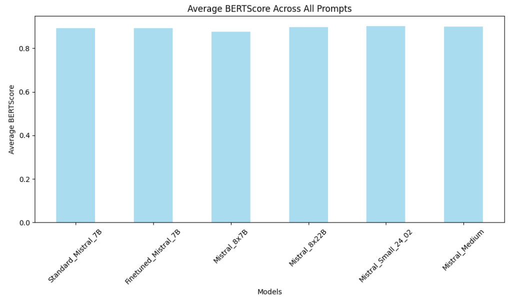
(a) ROUGE-1 Scores Comparison



(b) ROUGE-2 Scores Comparison



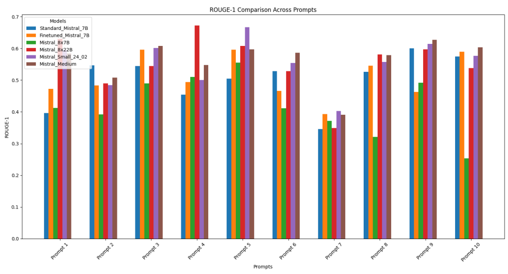
(c) ROUGE-L Scores Comparison



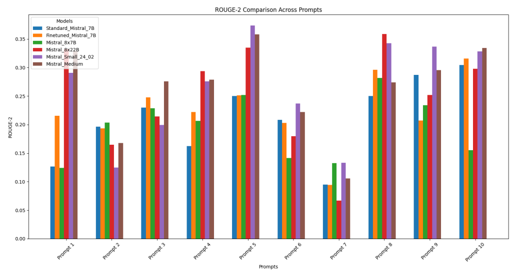
(d) BERTScore Comparison

Fig. 6.3: Model Score Comparisons by Metric

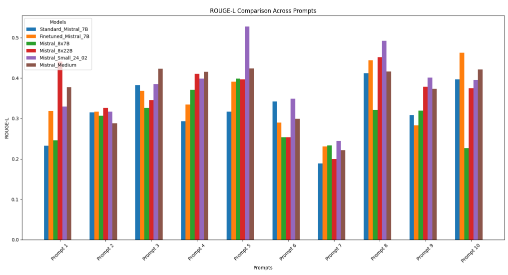
Scores achieved by particular Mistral models in each prompt can be seen in 6.4.



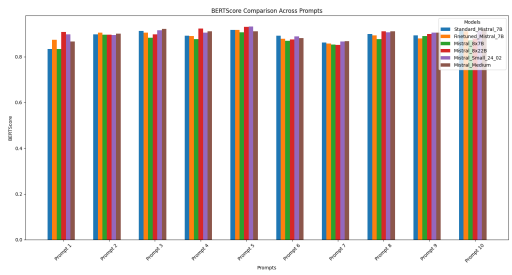
(a) ROUGE-1 Scores Comparison



(b) ROUGE-2 Scores Comparison



(c) ROUGE-L Scores Comparison



(d) BERTScore Comparison

Fig. 6.4: Model Score Comparisons by Metric

6.3. CASE STUDY 1: OPEN STACK LOGS

Log data 6.1 used in the first case study comes from Nova services of an OpenStack enviroment. There are three main activities that this log sample describes: Image Cache Operations, API Requests and Instance Resource Claim. Logs follow a structured format that consists of fields detailed below:

- Timestamp - Indicates when entry was generated in format YYYY-MM-DD HH:MM:SS .milliseconds
- Process/Thread ID - Identifies the process or thread responsible for the log entry (e.g. "2931")
- Log Level - Describes the severity or importance of the log entry (e.g. "INFO", "WARNING").
- Component or Service - Specifies the OpenStack component or service generating the log (e.g. "nova.virt.libvirt.imagecache").
- Context Information - Provides request context, usually tied to API calls or operations where "req-<UUID>" is unique identifier for api request.
- Message - Contains the main content of the log entry, detailing actions, errors, or warnings.
- Metadata - Includes additional details related to the log event (e.g. API response codes).

```
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:04:35.143 2931 WARNING nova.virt.libvirt.imagecache
↪  [req-addc1839-2ed5-4778-b57e-5854eb7b8b09 - - - - -] Unknown base file:
↪  /var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:04:35.144 2931 INFO nova.virt.libvirt.imagecache
↪  [req-addc1839-2ed5-4778-b57e-5854eb7b8b09 - - - - -] Removable base files:
↪  /var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:04:35.145 2931 INFO nova.virt.libvirt.imagecache
↪  [req-addc1839-2ed5-4778-b57e-5854eb7b8b09 - - - - -] Removing base or swap file:
↪  /var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742
nova-api.log.1.2017-05-16_13:53:08 2017-05-16 00:04:38.992 25746 INFO nova.osapi_compute.wsgi.server
↪  [req-d82fab16-60f8-4c9f-bde8-f362f57bdd40 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e - - -] 10.11.10.1
↪  "POST /v2/54fadb412c4e40cdbaed9335e4c35a9e/servers HTTP/1.1" status: 202 len: 733 time: 0.4953768
nova-api.log.1.2017-05-16_13:53:08 2017-05-16 00:04:39.184 25746 INFO nova.osapi_compute.wsgi.server
↪  [req-969a61db-496a-4350-8b5b-ff1bc11eb114 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e - - -] 10.11.10.1
↪  "GET /v2/54fadb412c4e40cdbaed9335e4c35a9e/servers/detail HTTP/1.1" status: 200 len: 1583 time: 0.1885760
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:04:39.301 2931 INFO nova.compute.claims
↪  [req-d82fab16-60f8-4c9f-bde8-f362f57bdd40 113d3a99c3da401fbd62cc2caa5b96d2 54fadb412c4e40cdbaed9335e4c35a9e - - -] [instance:
↪  ae3a1b5d-eec1-45bb-b76a-c59d83b1471f] Attempting claim: memory 2048 MB, disk 20 GB, vcpus 1 CPU
```

Listing 6.1: Raw Log Data (Prompt 1)

```
Key Log Activity:
- The provided logs contain various events such as system warnings, information messages, and instances creation/deletion activities.
- Notable activities include the removal of base or swap files at 2017-05-16 00:04:35 and instances creation at 2017-05-16 00:04:38
↪  and 00:04:39.

Errors and Incidents:
- Unknown base file warning at 2017-05-16 00:04:35 with log entry: nova.virt.libvirt.imagecache
↪  [req-addc1839-2ed5-4778-b57e-5854eb7b8b09 - - - - -] Unknown base file:
↪  /var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742
- This warning may indicate a potential issue with the system's image cache or base file management.

Security Recommendations:
- Investigate the `/var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742` file and determine its source and purpose.
- Implement regular system audits to monitor and identify potential security issues or anomalies in the image cache and base file
↪  management.
- Review and update access controls for the image cache and base file management systems to prevent unauthorized modifications or
↪  access.
- Implement logging and alerting mechanisms to notify administrators of any future warnings related to unknown base files.
- Regularly review and update system configurations to ensure they remain up-to-date and more secure.
```

Listing 6.2: Fine-Tuned Response (Prompt 1)

The fine-tuned model's response 6.2 offers a comprehensive and structured analysis of the provided logs, it focuses on the identification of key activities, incidents, and the provision of actionable recommendations. It highlights specific events such as the removal of base or swap files and the timestamps associated with instance creation. Each event is examined within the broader context of system operations.

Regarding errors and incidents, the fine-tuned response specifically pinpoints an unknown base file warning logged at 2017-05-16 00:04:35, directly linking it to potential issues in image cache management. This warning is identified as indicative of problems related to system configuration or base file handling.

The security recommendations provided by the fine-tuned response are precise and actionable. For instance, it suggests investigating the problematic file, conducting regular system audits, updating access controls, and implementing proactive monitoring with alerts for similar warnings. The response underscores the importance of maintaining up-to-date and secure configurations, thereby presenting a highly targeted and operationally focused approach. The primary audience for these recommendations includes system administrators and engineers seeking detailed guidance on resolving the issue and preventing its recurrence.

```
Key Log Activity:
- The logs contain multiple entries related to Nova Compute and Nova API services. Notable activities include instance creation with
↪  POST request, instance details retrieval with GET request, and removal of base or swap files.

Errors and Incidents:
- The log contains a WARNING message related to an unknown base file:
↪  /var/lib/nova/instances/_base/a489c868f0c37da93b76227c91bb03908ac0e742. This could potentially indicate a configuration issue or a
↪  missing file.

Security Recommendations:
- Regularly monitor logs for unexpected activities and warnings, as they may indicate potential security threats or system issues.
- Ensure that all system files are up-to-date and properly configured to avoid issues with unknown base files.
- Implement a backup and recovery solution for critical system files to minimize the impact of any potential data loss.
```

Listing 6.3: Standard Model Response (Prompt 1)

In contrast, the standard model's response offers a broader and less detailed overview of the logs. It summarizes key activities such as instance creation through a POST request, retrieval of instance details with a GET request, and the removal of base or swap files. However, it lacks specific timestamps and does not provide a detailed analysis of individual log entries. Regarding errors and incidents, the standard response mentions the unknown base file warning and suggests a potential configuration issue or missing file but fails to provide exact log entry details or timestamps.

The security recommendations provided by the standard model are more generalized. They include monitoring logs for unexpected activities, ensuring that system files are up-to-date, and implementing backup and recovery solutions. While these suggestions are useful, they lack the specificity required to address the particular issue identified in the logs. The standard model's response is more suited to offering a general overview for stakeholders who need a high-level understanding rather than detailed technical insights.

Table 6.1: Comparison of Fine-Tuned Response and Standard Model Response

Aspect	**Fine-Tuned Response**	**Standard Model Response**
Focus	Detailed, operational focus with specific timestamps and log entries.	High-level summary with generalized insights.
Error Analysis	Pinpoints errors with exact timestamps and log details, tying them to potential issues (e.g., image cache).	Broadly references errors without detailed context or log entry specifics.
Recommendations	Actionable and specific (e.g., investigate specific files, implement alerts for warnings).	Generic (e.g., monitor logs, update system files, and use backups).
Usability	Tailored for administrators or engineers who need clear next steps to resolve issues.	Suited for a general overview or reporting to less technical stakeholders.

6.4. CASE STUDY 2: LINUX LOGS

Second log sample 6.4 comes from Linux Operating System, in paricural SSH service running on it. These logs track SSH authentication attempts and other related activities. Structure of logs consists of:

- Timestamp – Indicates when entry was generated in format Month Day HH:MM:SS
- Host/System Name – Identifies the machine where the log was generated
- Process Name and ID – Identifies the process n format Process[Process ID]
- Log Message – Describes the event

Activity within the log sample consists of repeated authentication failures from the same IP's, attempts to use users like admin or matlab and pre-authentication disconnects.

```
Dec 10 10:14:04 LabSZ sshd[24833]: pam_unix(sshd:auth): check pass; user unknown
Dec 10 10:14:06 LabSZ sshd[24833]: Failed password for invalid user admin from 119.4.203.64 port 2191 ssh2
Dec 10 10:14:06 LabSZ sshd[24833]: pam_unix(sshd:auth): check pass; user unknown
Dec 10 10:14:08 LabSZ sshd[24833]: Failed password for invalid user admin from 119.4.203.64 port 2191 ssh2
Dec 10 10:14:08 LabSZ sshd[24833]: pam_unix(sshd:auth): check pass; user unknown
Dec 10 10:14:10 LabSZ sshd[24833]: Failed password for invalid user admin from 119.4.203.64 port 2191 ssh2
Dec 10 10:14:10 LabSZ sshd[24833]: pam_unix(sshd:auth): check pass; user unknown
Dec 10 10:14:13 LabSZ sshd[24833]: Failed password for invalid user admin from 119.4.203.64 port 2191 ssh2
Dec 10 10:14:13 LabSZ sshd[24833]: Disconnecting: Too many authentication failures for admin [preauth]
Dec 10 10:14:13 LabSZ sshd[24833]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=119.4.203.64
Dec 10 10:14:13 LabSZ sshd[24833]: PAM service(sshd) ignoring max retries; 6 > 3
Dec 10 10:19:59 LabSZ sshd[24839]: Connection closed by 1.237.174.253 [preauth]
Dec 10 10:21:01 LabSZ sshd[24841]: Invalid user matlab from 52.80.34.196
Dec 10 10:21:01 LabSZ sshd[24841]: input_userauth_request: invalid user matlab [preauth]
Dec 10 10:21:01 LabSZ sshd[24841]: pam_unix(sshd:auth): check pass; user unknown
Dec 10 10:21:01 LabSZ sshd[24841]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
↪ rhost=ec2-52-80-34-196.cn-north-1.compute.amazonaws.com.cn
Dec 10 10:21:09 LabSZ sshd[24841]: Failed password for invalid user matlab from 52.80.34.196 port 36060 ssh2
Dec 10 10:21:09 LabSZ sshd[24841]: Received disconnect from 52.80.34.196: 11: Bye Bye [preauth]
```

Listing 6.4: Raw Log Data (Prompt 10)

```
Key Log Activity:
- Multiple failed authentication attempts for user 'admin' from IP '119.4.203.64'.
- Connection closed from IP '1.237.174.253'.
- Invalid user 'matlab' attempt from IP '52.80.34.196'.

Errors and Incidents:
- Failed authentication attempts for invalid user 'admin' from IP '119.4.203.64'.
- Connection closed from IP '1.237.174.253'.
- Invalid user 'matlab' attempt from IP '52.80.34.196'.
- Non-standard port usage: port 2191 for ssh2 from IP '119.4.203.64'.

Security Recommendations:
- Implement a rate limiting system to block repeated login attempts from single IP addresses.
- Monitor and audit non-standard port usage and initiate investigations if breaches are detected.
- Review and update user permissions to ensure appropriate access levels.
- Implement multi-factor authentication for sensitive user accounts.
- Regularly review and update password policies, enforcing complexity and expiration.
- Implement log analysis tools to detect anomalous activities and suspicious patterns.
```

Listing 6.5: Fine-Tuned Response (Prompt 10)

The fine-tuned model's response delivers a comprehensive and technical analysis of the logs, concentrating on specific events, their implications, and providing targeted security recommendations. It identifies multiple failed authentication attempts for the user admin from the IP address 119.4.203.64, highlighting these incidents as potential brute-force activities. Additionally, it notes an invalid user login attempt for matlab from the IP address 52.80.34.196, suggesting another unauthorized access attempt. A key observation is the explicit mention of non-standard SSH port usage (port 2191) by 119.4.203.64, flagging it as an anomaly that might indicate probing attacks or misconfiguration.

In addressing errors and incidents, the fine-tuned response offers clear descriptions of failed login attempts, disconnections, and invalid user access attempts, each tied to specific IP addresses. It includes technical observations such as the use of non-standard ports, directly associating them with potential security risks.

The security recommendations provided by the fine-tuned model are actionable and specific. They include implementing rate limiting to block repeated login attempts, monitoring and auditing non-standard port usage, enforcing multi-factor authentication (MFA) for sensitive accounts, and regularly reviewing password policies. Additionally, it suggests utilizing log analysis tools to identify anomalies. These recommendations are precisely tailored to address the issues highlighted in the logs, making the response suitable for system administrators or security teams seeking detailed guidance.

```
Key Log Activity:
- Multiple failed login attempts for the user 'admin' from IP address 119.4.203.64, using SSH2 protocol on port 2191. The attempts
↪  were made at regular intervals.
- A failed login attempt for the user 'matlab' from IP address 52.80.34.196, also using SSH2 protocol.

Errors and Incidents:
- Repeated authentication failures for the user 'admin', indicating potential brute force attack or incorrect credentials.
- Unsuccessful login attempt for the user 'matlab', which could be a legitimate user or an attempt to gain unauthorized access.

Security Recommendations:
- Enable rate limiting on SSH authentication attempts to limit the number of failed login attempts within a specified time frame.
- Enforce strong password policies and consider implementing multi-factor authentication for users with elevated privileges, like
↪  'admin'.
- Monitor login attempts and configure intrusion detection systems to alert on unusual login patterns or brute force attacks.
- Consider implementing a system to lockout accounts after a certain number of failed login attempts to prevent unauthorized access.
- Review user accounts and ensure that only authorized users have access to the system.
- Regularly update and patch the system to address vulnerabilities.
```

Listing 6.6: Standard Model Response (Prompt 10)

In contrast, the standard model's response presents a broader and less detailed summary of the logs. It notes the repeated failed login attempts for the user admin from 119.4.203.64 and a failed login attempt for matlab from 52.80.34.196. However, the non-standard port usage (port 2191) is mentioned only as part of a general description without further analysis or emphasis on its significance.

Regarding errors and incidents, the standard model's response generalizes the repeated failed login attempts as either potential brute-force attacks or incorrect credentials. It does not delve into whether the failed matlab login attempt might indicate a probing attack or unauthorized access attempt.

The security recommendations in the standard model's response include generic best practices, such as enabling rate limiting for failed login attempts, enforcing strong password policies, and considering MFA for privileged accounts like admin. It also suggests monitoring login attempts, configuring intrusion detection systems, and locking accounts after multiple failed attempts. While these suggestions are valid, they lack specificity and are not tailored to the particular issues observed in the logs.

Table 6.2: Comparison of Fine-Tuned Response and Standard Model Response

Aspect	Fine-Tuned Response	Standard Model Response
Level of Detail	Detailed and technical, focuses on specific anomalies.	Generalized, summarizes events without diving into specific details or their implications.
Key Activity Focus	Highlights patterns and anomalies, including non-standard port usage and repeated failed attempts.	Focuses on broad descriptions of login attempts and patterns without analyzing anomalies deeply.
Recommendations	Actionable and specific.	Generic best practices applicable to many scenarios.
Audience	Suited for technical users who need to act on log details.	Suitable for non-technical audiences needing a general overview.
Port Usage Mention	Explicitly addresses and analyzes non-standard port usage as a potential risk.	Notes port usage but does not emphasize its significance or analyze it further.

In summary, the fine-tuned model's responses are detailed, operational, and tailored to a technical audience requiring actionable insights, such as system administrators or security teams. Conversely, the standard model's responses are offering a high-level overview more suitable for non-technical stakeholders, such as IT managers or compliance teams, focusing on general security practices rather than specific log-based insights.

6.5. LIMITATIONS AND OPPORTUNITIES

Fine tuned model excels in domain-specific tasks, however certain limitations should be taken into consideration. The model's performance is heavily dependent on the quality of the dataset used for fine-tuning, so logs from niche systems or environments not included in the training set may yield less accurate and more generalized outputs. Addressing this limitation requires continuous expansion and refinement of the dataset to encompass a broader range of log types and scenarios.

The computational efficiency of the fine-tuned Mistral 7B, while useful in small systems, poses challenges in scaling for larger datasets or real-time applications involving high-frequency log generation. Future research could explore hybrid models that combine the lightweight advantages of Mistral 7B with distributed processing frameworks to enhance scalability of the proposed solution. Fine-tuning the model specifically for anomaly detection could automate threat mitigation even further by bridging the gap between analysis and action.

7. CONCLUSIONS AND FUTURE WORK

The goal of this thesis was to demonstrate cybersecurity log analysis with artificial intelligence solution. Fine-tuned model for log summarization and anomaly detection was utilized, addressing the critical challenges posed by vast amounts of unstructured log data in modern systems. Model was integrated into a web application, enabling security analysts and system administrators to access fine-tuned model. Metrics such as ROUGE and BERTScore indicated increased performance of the fine-tuned model over the standard version, particularly in terms of accuracy and contextual relevance. These advancements not only reduce the burden on human analysts but could also significantly improve response times to security incidents.

Recognizing the critical role of logs in monitoring system activities, detecting anomalies, and mitigating threats, the Mistral 7B model was chosen for its balance between performance and computational efficiency, making it suitable for deployment even in resource-constrained environments. Fine-tuning involved creating a diverse dataset consisting of logs from Windows, Linux, Android, OpenStack, and Apache systems, ensuring the model's capability to generalize across different environments.

Case studies conducted on OpenStack and Linux logs highlighted the model's practical utility. For instance, it provided actionable insights by identifying unknown base file warnings in OpenStack logs and detecting repeated failed authentication attempts in Linux logs. Unlike the standard model, the fine-tuned version delivered operationally focused recommendations that could be directly implemented to enhance system security. Capabilities of the model were alos validated through metrics like ROUGE-2 and ROUGE-L, where the fine-tuned model demonstrated improvements of 13.8% and 4.52%, respectively.

To facilitate user interaction, a web application was developed using the Django framework, that consisted of modules for log summarization, question answering, and previous conversations storage. The integration of the Mistral API ensured efficient communication with the language model, while a Bootstrap-based frontend enhanced user experience.

Despite these advancements, certain limitations were noted. The fine-tuned model's performance is heavily influenced by the quality and diversity of its training dataset. Logs from niche systems or environments not included in the training set may yield less accurate results. Additionally, real-time data processing capabilities and a broader variety of log formats remain areas for future development.

In conclusion, this thesis highlights the potential of integrating transformer-based Large Language Models into cybersecurity practices. The fine-tuned Mistral 7B model demonstrated how AI can automate complex tasks, offering greater accuracy and speed compared to manual methods. By simplifying log summarization and anomaly detection, this approach enables quicker responses to potential security incidents. Future research should explore models with enhanced reasoning capabilities, support for more log formats, and the ability to detect sophisticated types of attacks, further advancing the field of AI based cybersecurity solutions.

BIBLIOGRAPHY

[1] Balasubramanian, P., Seby, J., Kostakos, P., *Cygent: A cybersecurity conversational agent with log summarization powered by gpt-3*. 2024.

[2] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D., *Language models are few-shot learners*, CoRR. 2020, tom abs/2005.14165.

[3] Chuvakin, A.A., Schmidt, K., Phillips, C., *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management* (2012).

[4] Devlin, J., Chang, M., Lee, K., Toutanova, K., *BERT: pre-training of deep bidirectional transformers for language understanding*, CoRR. 2018, tom abs/1810.04805.

[5] Foundation, D.S., *Django: The web framework for perfectionists with deadlines*. 2024. Accessed: 2024-10-12.

[6] Franco, M.F., Rodrigues, B., Scheid, E.J., Jacobs, A.S., Killer, C., Granville, L.Z., Stiller, B., *Secbot: a business-driven conversational agent for cybersecurity planning and management* (IFIP, 2020), str. 1–7.

[7] Garcia, K.A., Monroy, R., Trejo, L.A., Mex-Perera, C., Aguirre, E., *Analyzing log files for postmortem intrusion detection*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2012, tom 42, 6, str. 1690–1704.

[8] Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E., *Mistral 7b*. 2023.

[9] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., *Survey of intrusion detection systems: techniques, datasets and challenges*, Cybersecurity. 2019, tom 2, 1, str. 1–22.

[10] Kwon, H.Y., Kim, T., Lee, M.K., *Advanced intrusion detection combining signature-based and behavior-based detection methods*, Electronics. 2022, tom 11, 6, str. 867.

[11] Leite, J.P., *Analysis of log files as a security aid*, w: *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)* (2011), str. 1–6.

[12] Li, H., Chen, Y., Luo, J., Kang, Y., Zhang, X., Hu, Q., Chan, C., Song, Y., *Privacy in large language models: Attacks, defenses and future directions*, arXiv.org. 2023, tom abs/2310.10383.

[13] Lin, C.Y., *Rouge: A package for automatic evaluation of summaries*, w: *Text summarization branches out* (2004), str. 74–81.

[14] McIntosh, T.R., Liu, T., Sušnjak, T., Alavizadeh, H., Ng, A.H.M., Nowrozy, R., Watters, P.A., *Harnessing gpt-4 for generation of cybersecurity grc policies: A focus on ransomware attack mitigation*, Computers & Security. 2023, tom 134, str. 103424–103424.

[15] Mistral AI, *Mistral ai documentation*. 2024. Accessed: 2024-10-31.

[16] Morgan, S., *Cybercrime to cost the world $10.5 trillion annually by 2025*, Cybersecurity Ventures. 2020. Accessed: 2024-12-01.

[17] Mudgal, P., Wouhaybi, R., *An assessment of chatgpt on log data*, arXiv.org. 2023, tom abs/2309.07938.

[18] New Relic, *2022 state of logs*. 2022. Accessed: 2024-12-01.

[19] Przybyszewski, T., *Thesis project repository*, `https://github.com/tprzybyszewski/thesis`. 2024. Online access: accessed on December 5, 2024.

[20] Qi, J., Huang, S., Luan, Z., Fung, C., Qian, D., *Loggpt: Exploring chatgpt for log-based anomaly detection*, arXiv.org. 2023, tom abs/2309.01189.

[21] Radford, A., *Improving language understanding by generative pre-training*. 2018.

[22] Raschka, S., Liu, Y.H., Mirjalili, V., *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python* (Packt Publishing Ltd, 2022).

[23] Russell, S., Norvig, P., *Artificial Intelligence: A Modern Approach (4th Edition)* (Pearson, 2020).

[24] Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., Gadepally, V., *From words to watts: Benchmarking the energy costs of large language model inference*. 2023.

[25] Shrager, J., *Eliza reinterpreted: The world's first chatbot was not intended as a chatbot at all*. 2024.

[26] Team, S.D., *Sqlite - self-contained, high-reliability, embedded, full-featured, public-domain, sql database engine*. 2024. Accessed: 2024-11-26.

[27] Team, T.B., *Bootstrap - the most popular html, css, and js library in the world*. 2024. Accessed: 2024-11-26.

[28] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G., *Llama: Open and efficient foundation language models*. 2023.

[29] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T., *Llama 2: Open foundation and fine-tuned chat models*. 2023.

[30] Tunstall, L., Von Werra, L., Wolf, T., *Natural language processing with transformers* (" O'Reilly Media, Inc.", 2022).

[31] Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y., *Bertscore: Evaluating text generation with bert*. 2019.

[32] Zhu, J., He, S., He, P., Liu, J., Lyu, M.R., *Loghub: A large collection of system log datasets for ai-driven log analytics*. 2023.