Leveraging Large Language Models for Network Traffic Analysis: Design, Implementation, and Evaluation of an LLM-Powered System for Cyber Incident Reconstruction

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Cyber Security
November 2024
Md Naeemur Rahman

Supervisors:
Tahir Mohammad (University of Turku)
Seppo Virtanen (University of Turku)

UNIVERSITY OF TURKU
Department of Computing

Md Naeemur Rahman: Leveraging Large Language Models for Network Traffic Analysis: Design, Implementation, and Evaluation of an LLM-Powered System for Cyber Incident Reconstruction

Master of Science (Tech) Thesis, 70 p.
Cyber Security
November 2024

Cyberthreats are evolving, and becoming more sophisticated. Hence, there is a growing need for advanced security analysis tools. Traditional approaches for analyzing network traffic and reconstructing cyber incidents struggle to efficiently process and analyse large amounts of network data in real-time. The rapid advancement of Generative AI, particularly large language models (LLMs), opens up new possibilities for extending current digital forensics and incident response capabilities. LLMs can play an important role in cybersecurity, but their adoption has been limited due to reliance on commercial models, data privacy concerns, and a lack of domain-specific fine-tuning.

This thesis proposes a new type of framework that uses locally run open-source LLMs and vector databases to perform network traffic analytics to reconstruct a cyber incident and identify the attacker. The architecture of the proposed system includes five major components: (1) a data preprocessing pipeline that ingests the network packet capture (PCAP) files and extracts relevant features while also using external threat intelligence data source, such as "VirusTotal", to provide a context-aware response; (2) a vector database (ChromaDB) that stores the preprocessed data for fast similarity search and retrieval; (3) Locally run open-source LLMs (LLAMA, Falcon, and Mistral) which analyze the retrieved data and generate human-like responses to the security queries; and (4) a user interface that allows security analysts to interact with the system, to gather insights.

The proposed system was tested using real-world network traffic data and offers a practical framework for using advanced LLM in network security analysis while ensuring data privacy and system efficiency. The comparison of LLMs like LLAMA, Falcon, and Mistral with different parameter sizes (7B,13B) provides useful insights for selecting and optimizing models for real-world use. The results indicate that these open-source LLMs have great potential to provide accurate and smart responses to security queries. The results indicate that larger models (large parameters) generally perform better than smaller ones in accuracy. Llama7-13b stood out as the top performer across all metrics. Furthermore, designing a detailed form with sufficient information and context is important to guide the output of LLMs and obtain relevant insights.

Keywords: Large Language Models, Cybersecurity, Network Traffic Analysis, Digital Forensics, Artificial Intelligence, Incident Response

Contents

List of Figures

List of Tables

List of acronyms

AI Artificial Intelligence

API Application Programming Interface

APT Advanced Persistent Threats

BERT Bidirectional Encoder Representations from Transformers

CWE Common weakness enumerations

GPT Generative Pre-trained Transformers

LLMs Large Language Models

ML Machine Learning

NER Named Entity Recognition

NIST National Institute of Standards and Technology

NLP Natural Language Processing

NTA Network traffic analysis

RTID Robust Transformer-based Intrusion Detection System

SIEM Security Information and Event Management

T5 Text-To-Text Transfer Transformers

1 Introduction

We are living in an interconnected digital world where cyberattacks are not only becoming more frequent but also more advanced and harder to detect. It has become a key area of concern as Generative Artificial Intelligence started to take off. This widespread use of Artificial Intelligence (AI) has lowered the entry barrier for cybercriminals, expanding the threat landscape and putting more systems at risk. Traditional cybersecurity methods such as threat detection and incident response systems are struggling to keep up with the sheer number of attacks [1] [2]. With the continuous generation of network traffic and data, manual monitoring is not an option anymore, making dangerous critical infrastructures and businesses vulnerable to breaches. The increasing sophistication of cyberthreats, including malware, ransomware, and Advanced Persistent Threats (APTs), demands robust defensive strategies [3].

Digital forensics plays a crucial role in modern cybersecurity strategies. It involves identifying, recovering, and investigating digital evidence systematically from electronic devices. This not only aids in detecting security breaches but also supports post-incident investigations, enabling organizations to understand the full scope of an attack, mitigate its damage, and potentially pursue legal action against perpetrators. With the surge in the number of networked devices and the complexity of modern cyberattacks, network traffic analysis has emerged as a critical component of digital forensics. It allows analysts to traverse forward, interpret, and reconstruct the timeline of events in an attack to help identify exploitable vectors and also trace the sources of malicious activity.

With cyberthreats growing more complex and sophisticated all the time, the need for modern investigation techniques arises. This is where the importance of AI comes in, which provides revolutionary approaches to streamline digital forensic workflows. This thesis explores the potential of utilizing Large Language Models (LLMs) in forensic investigations to improve efficiency. Combining the capabilities of AI and LLM, digital forensic experts can not only speed up the identification and analysis of possible threats in real-time but also improve their findings.

1.1 Emergence of Large Language Models

Conventional digital forensics is highly dependent on manual analysis and rules-based systems, which are proving to be less practical as they are less effective for the current levels of cybersecurity threats. Hence, a paradigm shift toward automated and LLM-powered digital forensics is underway and is expected to greatly enhance the capabilities to accomplish incident reconstruction and response. That being said, the emergence of automated and AI forensics can be a radical evolution of the digital forensics field and is a major step forward in the incident reconstruction and response process. LLMs can analyze large amounts of unstructured data such as network traffic, threat intelligence, reports, and user behavior patterns to find potential threats and anomalies in security [4]. Because LLMs are good at understanding context and semantics, they can identify subtle patterns and correlations that rule-based systems may not detect; these very capabilities also make them especially suited to hunting new and as-yet unseen threats such as zero-day vulnerabilities and advanced persistent threats. For example, Fu *et al.* [5] introduced LLM4SecHW, a framework that fine-tunes mid-sized LLMs with datasets of hardware design defects and test the performance of that model for identifying bugs in hardware systems.

This innovative approach provides a blueprint for integrating domain-specific LLMs into other fields. By combining AI and large language models, there is an immense potential to enhance and automate network traffic analysis for digital forensics.

1.2 Research problem and its significance

Traditional forensic methodologies, though effective, often require manual intervention and are reactive by nature. As cyberthreats become more sophisticated, timely and accurate incident reconstruction becomes harder. Traditional approaches are often slow and cannot process large volumes of network traffic efficiently. As a result, their usefulness in real-time attack detection and response is limited. One of the major limitations of current traffic analysis methods for digital forensics methods is their reliance on static analysis techniques and traditional rule-based systems, which struggle to adapt to the dynamic nature of modern cyberattacks [6]. Moreover, existing digital forensics tools may fail to fully extract and interpret subtle patterns within large datasets, leaving gaps in incident response strategies. The demand for more advanced systems that can analyze massive datasets, detect patterns indicative of cyberthreats, and help reconstruct cyber incidents is more pressing than ever.

This thesis will address these limitations involving traditional and ML-based triage methods by introducing Generative AI-powered systems that leverage LLMs to analyze large amounts of network traffic data to enhance the capabilities of cyber incident reconstruction. Furthermore, the accuracy of digital forensics investigations can be greatly enhanced by using LLMs. However, while their integration offers significant opportunities to streamline and enhance forensic processes, challenges and limitations remain. Challenges like handling large amounts of data and combining different types of evidence must be solved to fully benefit from these advanced technologies. Addressing these issues is key to maximising LLMs in digital forensics.

1.3 Research Questions

Based on the research problem presented, this thesis aims to find answers to the following research questions:

1. *Research Question 1 (RQ1)*: What are the challenges in conventional approaches for incident response and cyber incident reconstruction, and how can the LLM assist?

2. *Research Question 2 (RQ2)*: What LLMs models be used to assist in the cyber incident reconstruction that are cost-effective and can be customized?

3. *Research Question 3 (RQ3)*: How effective are the LLM-assisted in cybersecurity incident reconstruction analysis?

1.4 Research Objective

The primary objective of this research is to develop and evaluate a GenerativeAI-powered system that leverages LLMs to analyse network traffic data to enhance digital forensics investigations. The system aims to assist in reconstructing cyber incidents by identifying patterns within log data and digital artifacts, providing forensic analysts with deeper insights into the structure and method of attacks. In addition to this, this thesis work is meant to respond to some of the key gaps in current approaches for digital forensic materials and network traffic analysis objectives. The thesis centres on the following objectives:

1. Investigate and study inherent problems in traditional digital forensics methodologies from the perspective of managing large and complex system network traffic data.

2. Explore the capabilities of Locally run LLMs and provide a framework for processing and analyzing large volumes of network traffic data and log files.

3. Compare the effectiveness, speed, and scalability of LLM-powered traffic analysis (as run locally) to traditional ones.

1.5 Contributions

The key contributions of this research are:

1. Development of a Generative AI-powered Network Traffic Analyzer: This system will use locally operated LLMs for analyzing network traffic and log file data to be produced. In itself an attack pattern recognizer, it is also to investigators about the manner in which their assailants work.

2. Evaluation of Open-Source Models: Present the evaluation of open-source models like LLAMA, Falcon and Mistral in network traffic analysis regardless of their size or number of parameters

3. Practical Guidelines for Implementation: The research will offer guidelines for implementing LLM-based digital forensics systems in real-world cybersecurity scenarios, addressing issues such as data privacy and the admissibility of digital evidence in legal contexts.

1.6 Structure of the Thesis

The rest of the thesis is organized as follows:

Chapter 2: Literature Review - This chapter introduces Generative AI and LLMs and reviews their architecture, training approaches, and current applications. It surveys the current research trends on employing LLMs in cybersecurity. It also reviews current methodologies and notes the gaps in the literature that this thesis will address.

Chapter 3: Methodology - This chapter provides the research methodology of this work. It includes the design of the LLM-powered system. It provides a detailed description of the proposed method for network traffic analysis. It also explains the proposed system's workflow, how LLMs are integrated, and the role of LLMs.

Chapter 4: System Design and Implementation - In this chapter, the technical details of the proposed system are described. It also includes the design and setup of the actual system. It also provides a detailed description of each component of the system and how it works.

Chapter 5: Testing and Evaluation - This chapter discusses the outcome of the system evaluation. It involves dataset selection, performance analysis, and case studies.

Chapter 6: Conclusion and Future Work - The final chapter summarizes the remaining findings from this research and also revisits the research objectives to show how they have been met. It also includes a look at where this research could be headed.

2 Literature Review

This literature review chapter aims to provide a comprehensive overview of current research on applying LLMs in various cybersecurity areas. This section will explore how these powerful models are being utilized to improve threat detection, vulnerability assessment, and the overall security posture of digital environments. The review chapter will cover several key areas, which are:

- Background of Generative AI.
- The evolution and current state of LLMs
- Applications of LLMs across different cybersecurity domains
- Challenges and limitations in applying LLMs to cybersecurity tasks
- Emerging trends and future directions in LLM-based cybersecurity solutions

By reviewing recent scientific literature and examining these areas, this chapter aims to provide solid background knowledge for understanding the potential of LLMs in cybersecurity practices and identify gaps in current research.

2.1 Background

2.1.1 Generative AI and LLM

Generative AI is a term that describes a category of AI models that are designed to generate new data instances from what they were trained on. Generative AI is important in applications such as Natural Language Processing (NLP) and content generation because it is capable of generating meaningful contextual output. The most visible form of generative AI at scale comes in the form of LLMs, which are trained to generate coherent text as produced by humans. These models are based on deep learning architectures, and they are trained on thousands of terabytes of data, enabling them to achieve state-of-the-art performance across a wide variety of natural language processing tasks [7].

Thus, they are able to perform tasks that require anything from generating simple text to performing sophisticated reasoning and even analysis [8]. Unlike traditional neural networks, which process data sequentially, LLMs simultaneously process multiple data elements, allowing LLMs like Generative Pre-trained Transformers (GPT), Bidirectional Encoder Representations from Transformers (BERT), and Text-To-Text Transfer Transformers (T5) to process and understand complex language patterns. These models are purpose-built to process various linguistic tasks, from machine translation to summarization and question-answering.

2.1.1.1 Historical Context and Core Technologies

Language processing has evolved from classical rule-based systems and statistical methods to deep learning techniques and advances with Generative AI and LLMs. The availability of computational power and access to large datasets has made this transition possible. Such resources and data have allowed for the training of complex models that capture the intricacies of human language.

In this domain, one of the most important innovations is the introduction of transformer architecture, which drastically improved traditional natural language processing methods and became the state-of-the-art of large-scale transformer-based models. NLP was used in cybersecurity mostly for basic text analysis and pattern matching in the early days. But, it was the transformer architecture that was a semantic boundary in the field. Vawani *et al.* [9] in 2017 introduced the transformer model that revolutionized the field of NLP by enabling parallel processing of sequential data and significantly improving performance on various language tasks. This architecture laid the foundation for developing large-scale language models that have become increasingly relevant to cybersecurity applications. The transformer architecture, which revolutionized natural language processing through several key mechanisms:

- Attention Mechanisms: These allow models to weigh the importance of different parts of input data dynamically, as a result, the model understands the context better [9] [10].

- Self-Attention: This mechanism enables models to understand relationships between different parts of the input text, which improves the comprehension and generation capabilities of the model.

- Parallel Processing: Unlike earlier sequential models, transformers can process input data in parallel. This approach is much faster than earlier sequential models, and as a result, it greatly increases efficiency. [10]

The capabilities of language models improved significantly with the release of models like BERT [11] and GPT series [12] [13]. These models excelled at a variety of natural language processing (NLP) tasks, including common cybersecurity tasks such as question answering, named entity recognition, and text classification. According to Xu *et al.* [7], the use of LLMs in cybersecurity has risen dramatically since 2020. This increase is due to the models' ability to process and comprehend large amounts of textual data, which is critical for cybersecurity because natural language is frequently used to describe threat intelligence, vulnerability descriptions, and attack patterns.

2.1.1.2 How LLMs Work and Importance of LLMs

LLMs are powerful because of their advanced training and wide-ranging abilities. LLMs are quickly becoming a cornerstone of modern tech, especially around cybersecurity and digital forensics.

2.1.1.3 Training Process

LLMs are based on a complex and resource-intensive training process. The training process consists primarily of feeding the model large datasets in the form of text data so that it can learn patterns, relationships, and context from the dataset. As stated by Ferrag *et al.* [14], pre-training frequently utilizes a large text data set of hundreds of billions of tokens from various sources. During the pre-training step, the model encodes into an understanding of language in general and obtains the relevant analytic ability, such as predicting masked words or estimating the likelihood of continuing the tokens in history.

After the pre-training phase, models are fine-tuned based on a smaller dataset. This is a very important phase, allowing them to fine-tune to certain domains or tasks. This further training occurs on carefully curated datasets, so these systems are guided by domain-specific data. As noted by Myers *et al.* [8], fine-tuning enables models to retain their broad language understanding while gaining tailored capabilities for specific tasks. This two-stage approach (pre-training, then fine-tuning) has also shown great success in constructing models that perform well in terms of a broad understanding of context and performing well on specific tasks. The diagram in Figure 2.1 illustrates the pre-training and fine-tuning phases of LLM.

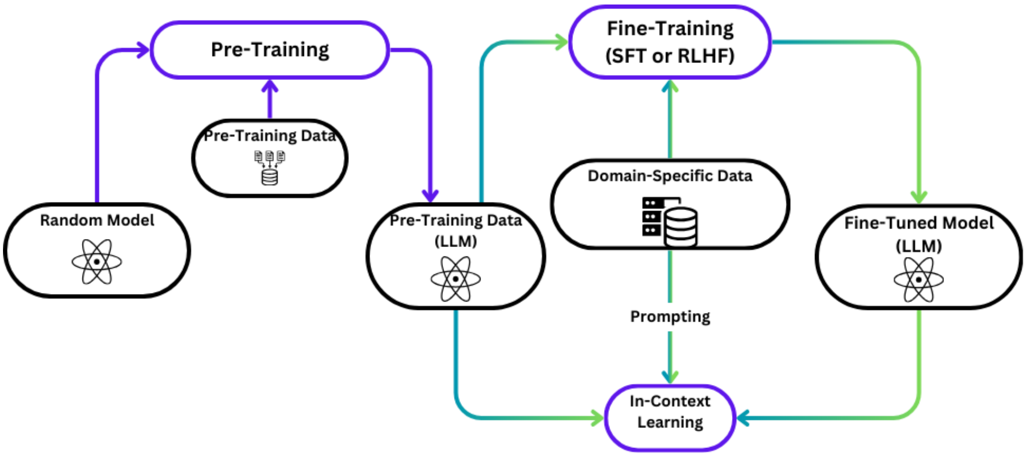

Figure 2.1: LLM Pre-training, Fine-Tuning, and In-Context Learning Phases [15]

Tokenization is a vital part of how the LLM works. The text input is recursively split into tokens, which can be words, subwords or characters depending on the used tokenization scheme. Tokenization stands as one of the essential processes for LLM functioning. It is the process of splitting the input text systematically into tokens, and tokens are the basic unit of the text that can mean words, sub-words, or characters depending on the type of tokenization used. Xu *et al.* [7] highlights that improved tokenization techniques allow models to better understand languages and different text formats, facilitating them to be more versatile and widely used.

2.1.1.4 Capabilities of LLMs and Relevance to Cybersecurity

According to Yao *et al.* [16], modern LLMs possess sophisticated reasoning skills that enable them to comprehend intricate queries, find pertinent information in data, and preserve context over lengthy text passages. These models are very adept at producing coherent, contextually relevant text with long-range consistency across an output. They can adapt extremely fast to different kinds of styles and domain needs. Their strength in transferring knowledge between fields and adapting to new tasks with little additional training makes them ideal for rapidly developing areas like cybersecurity. Qiu *et al.* [17] noted that this flexibility enables the LLMs to synthesize information from several sources and make detailed analyses and inferences that are difficult to achieve through conventional systems.

Leveraging LLMs in Cybersecurity marking a paradigm shift in organizations' approach to threat detection, analysis, and response. These models provide accuracy along with huge opportunities to handle and analyze the huge amount of data from the sources of modern systems. As concluded by various authors' studies in the area of cybersecurity [18] [10], LLMs are capable of exemplary real-time threat detection by tracking network traffic in real-time and detecting possible security attacks with high precision. Their ability to understand context helps reduce false positive alerts, which is a common challenge in traditional security systems.

Moreover, LLMs have already shown remarkable effectiveness in processing massive volumes of security log data during automated analytics processing. Yao *et al.* [16] showed that such custom detection methods could even discover attack patterns and anomalies in the data that are not easily recognizable through traditional detection methods. Such LLMs can produce specific security reports and recommendations to enable security teams to respond to incidents more efficiently. Additionally, these models have automation capabilities that simplify repetitive security tasks, enabling human analysts to devote their time and energy to complex challenges.

In addition, LLMs can significantly enhance incident response through rapid evaluation of security events and the generation of response plans. They are capable of rapidly absorbing past incidents and adapting to new variants of threats enabling them to be power tools against the evolving world of cyberthreats. As Ferrag *et al.* [14] stated, such adaptability is indispensable when countering zero-day attacks and novel threat vectors typically not recognized by conventional detection systems.

2.1.2 Incident Response

Cybersecurity incident response is a process used to help locate, isolate, and eliminate threats to informational systems. It hopes to minimize damage and maintain low costs of recovery. According to Cichonski *et al.* [19], incident response is a structured organization of the steps to be taken regarding security breaches. It has a mix of technical action and organizational effort to create a robust defense. On top of that, it also encompasses policy-making, team organizing, and organization-wide strategic decision-making. According to the National Institute of Standards and Technology (NIST), incident response is a set of actions taken to respond to a violation of security policies [19]. This definition clearly indicates that there is a requirement for standardized processes and procedures to respond to such security events. As organizations experience ever more frequent and complex cyber-attacks, having these frameworks is more important now than ever.

2.1.2.1 Traditional Approach

Traditional incident response systems started from basic computer security practices. They developed into formal structures as cyberthreats became more common. These systems use a structured approach to manage incidents and include various tools and methods to detect and respond to security issues. Cichonski *et al.* [19] explain that traditional frameworks rely heavily on set procedures and human expertise. They often follow guidelines and best practices built over many years of cybersecurity work. According to Ahmad *et al.* [20], traditional incident response systems originated from fundamental computer security practices and evolved into formal structures as cyberthreats became more prevalent. The author also emphasized traditional frameworks rely heavily on established procedures and human expertise.

2.1.2.2 How Traditional Incident Response Systems Work

Traditional incident response systems operate through a series of well-defined phases and methodologies. It follows clear steps and involves multiple phases. According to a study by Sumanth Rao [21], these systems focus on detailed investigation and response. They use a mix of automated tools and manual analysis to find and handle security incidents. This approach has proven effective in dealing with known patterns of threats, but it faces major challenges when confronted with novel or sophisticated forms of attack vectors. A survey study by Alzaabi *et al.* [22] shows that traditional systems mainly use rule-based detection and signature analysis. These methods are good at spotting familiar threats but perform poorly on detecting zero-day attacks or advanced techniques that do not match known patterns. Traditional systems also depend on human analysts to make decisions and link incidents, which is challenging when dealing with complex, multi-layered attacks.

2.1.2.3 Phases of Incident Response

Traditional incident response systems follow a clear structure with several key phases. According to NIST Special Publication by Cichonski *et al.* [19], this process starts with preparation and moves through detection, analysis, containment, eradication, and recovery. Each phase is vital to processing security incidents and lays a very important foundation for response to breaches. Preparation is the cornerstone of any effective incident response. A persistence case study by Ahmed *et al* [20] shows good results for organizations that have invested heavily in preparation for real incidents. This includes creating incident response plans, forming response teams, and running regular training exercises. This preparatory work creates the necessary infrastructure for managing security events effectively. This finding is further supported by Verizon's 2023 "Data Breach Investigations Report"(DBIR) [23], which analyzed multiple case studies showing that organizations investing in thorough preparation—such as developing incident response plans, forming dedicated teams, and conducting regular training exercises—were able to detect incidents more quickly and respond more effectively to security breaches.

Detection and analysis are the first steps in responding to potential security incidents. These are when security teams identify and look into possible threats. Alzaabi et al. [22] explain that traditional detection relies on tools like Security Information and Event Management (SIEM) systems, Intrusion Detection Systems (IDS), and log analysis software. These tools create alerts based on set rules and signatures. Security analysts then need to check these alerts to see if they are real threats and how severe they might be.

The containment phase is about stopping confirmed security incidents from spreading further. According to Bartnes *et al.* [24] traditional strategies include isolating affected systems, applying temporary fixes, and saving evidence for deeper analysis. Their case study focused on Norwegian electric power companies which demonstrates that this phase requires careful balance between maintaining business continuity and preventing further spread of the security incident. Another case study by Ahmad *et al.* [20] also emphasized that effective containment strategies must consider both technical and business impact factors when responding to security incidents.

2.1.2.4 Tools and Technologies

Traditional incident response systems use many tools and technologies to work effectively. According to [25], several main types of tools are used in these systems which are:

- *Network Monitoring Tools:* Network monitoring tools are crucial for finding and analyzing suspicious activity. These tools are used to look at network traffic, system logs, and application behavior to spot possible security incidents.

However, their success often depends on the quality of the detection rules and signatures they use. Hofstede *et al.* [26] provide an in-depth explanation of flow monitoring, including packet capture and data analysis using NetFlow and IPFIX. The author also points out that effective network monitoring requires a combination of packet analysis, flow monitoring, and behavioral analytics.

- *Security Information and Event Management (SIEM) Systems:* SIEM systems are like the central nervous system for traditional incident response. These platforms collect and connect data from different sources. This helps security teams see all potential security incidents in one place. According to a study by Cinque *et al.* [27], SIEM systems encounter significant challenges in processing the volume and variety of data produced by modern networks. The research highlights that the effectiveness of incident response heavily depends on the integration of various security tools and monitoring systems working in concert to detect and respond to threats.

- *Role of Human Analysts:* Human analysts are central to traditional incident response. They make key decisions at every step of incident management. Sundaramurthy *et al.* [28] highlights that human expertise is vital for interpreting alerts, investigating issues, and choosing the right response. However, depending on human judgment has both advantages and drawbacks for the response process.

2.1.2.5 Challenges in Traditional Systems

Modern cybersecurity landscapes present many challenges that strain traditional incident response capabilities. Freitas and Gharib [29] point out several major weaknesses in these older systems. One of the biggest issues is their ability to scale as data volumes and attack methods grow more complex. Their research highlights how these traditional methods struggle to maintain performance and accuracy when processing large amounts of data across complex enterprise environments. Scalability is another primary concern in traditional incident response systems. Scalability is another major concern for incident correlation systems. Organizations are producing more and more security data at an exponential rate. As a result, conventional analysis methods are struggling to keep up with this overwhelming amount of data. Freitas and Gharib [29] also illustrated in their studies how conventional systems struggle to efficiently correlate billions of alerts, leading to potential delays in the timely detection and response to security incidents.

Another study by González-Granadillo *et al.* [30] shows that traditional systems struggle with performance when processing large amounts of data. This study highlights that SIEM systems encounter significant challenges in processing a large volume of data, which leads to delays in detecting and responding to incidents. Another major issue is the heavy reliance on human analysis. According to Sundaramurthy *et al.* [28], the heavy reliance on human analysts for interpreting alerts and investigating incidents can introduce delays, especially during large or complex incidents. This dependence can slow down response times and lead to inconsistencies in incident handling. Traditional systems also have trouble managing the complexity of modern cyberthreats. The research by Wang *et al.* [31] highlights that conventional incident response frameworks may fail to identify and address sophisticated attack patterns that utilize multiple attack vectors simultaneously. This study emphasizes that reconstructing lateral movements and understanding the full scope of multi-vector APT attacks requires advanced, integrated approaches to improve detection and response capabilities

2.1.3 Cyber Incident Reconstruction

Cyber incident reconstruction is an important part of digital forensics and incident response. It helps organizations understand the full scope and impact of security incidents. According to Casey [32] incident reconstruction means putting together different digital artifacts and evidence to build a clear timeline of events before, during, and after an incident. This process is essential for quick incident response and for making long-term security improvements.

2.1.3.1 Importance of Network Traffic Analysis

An important component of incident reconstruction is Network traffic analysis (NTA). It provides critical insight into how attacks are carried out and the methods used by the attacker. According to Thakare *et al.* [33], NTA involves the "capturing of network traffic and inspection of it closely enough to understand what is happening network". This method converts the data packets and displays them in plain text. Network Traffic Analysis is used for various purposes, including proactive network optimization and security. Nour Alqudah *et al.* [34] described an overview of where traffic analysis is commonly used, for example, assessing use case scenarios, efficiency and security of network governance and performance. NTA is considered as important for the security as well as the performance of a network. As network traffic increases, new ways are needed to detect intrusions, classify Internet traffic, and analyze virus behavior.

Dhakad *et al.* [35] states that modern network traffic is both extremely fast and a large volume, complicating analysis. The rise in encrypted traffic has also created new challenges for traditional methods. Nowadays, modern network traffic analysis methods use Machine Learning (ML) and AI. These methods "utilize machine learning to learn and recognize malicious traffic patterns" and are "more potent against emerging and changing threats" [35]. This is crucial because traditional signature-based detection is not always effective against new or unknown threats. Modern network traffic is studied at different levels, as described by Thakare *et al* [33].

- Flow Level Analysis: Examines features like "duration of the flow, volume of data, number of packets per flow". Flow analysis is useful for spotting command-and-control traffic and attempts to steal data. However, it needs a lot of computing power for real-time processing.

- Packet Level Analysis: Studies characteristics such as "length of the packet, mean and variance of the packet length"
- Network Level Analysis: Looks at elements like hostnames and server names

The challenges in network traffic analysis are significant. This thesis will explore LLM-based solutions for analyzing network traffic, which would enable organizations to understand network behavior better, detect anomalies, and respond to security incidents more effectively.

2.2 Current State of LLMs in Cybersecurity

LLMs such as GPT-3, GPT-4, LLAMA, Falcon LLM, and BERT have brought major changes to various industries, including cybersecurity. These models can analyze large data sets, adapt to different situations, and handle complex tasks without the need for specific programming. The use of LLM in cybersecurity has altered how security challenges are addressed. Recent research indicates an increasing use of LLMs for security tasks, such as vulnerability detection and threat intelligence analysis [7]. The rapid development of LLMs, particularly open-source models such as LLAMA3, Falcon, and Mistral, has resulted in new ways to automate and improve cybersecurity operations. In recent years, LLMs have demonstrated remarkable abilities to comprehend and process complex security-related tasks. Ferrag *et al.* [14] found that LLMs have significant potential in several areas, including:

- Automated vulnerability detection
- Malware analysis

- Network intrusion detection
- Phishing detection
- Threat intelligence generation

This has driven the field to develop sophisticated techniques for security analysis and threat detection that have grown progressively automated. [14] Moreover, there is rising interest in devising ways to make LLMs more efficient and accessible for deployment in resource-constrained environments. That is essential for real-time cybersecurity operation [36]. Model compression, knowledge distillation, and efficient fine-tuning are some of the techniques being investigated to make LLMs more amenable to these tasks.

2.2.1 Vulnerability Detection and Analysis

Vulnerability detection and analysis is one of the most common applications of LLMs in cybersecurity. Current approaches are mostly based on matching regular expression patterns from the source code, such as from the OWASP vulnerability patterns. LLMs already have already shown great capabilities to enumerate and analyze vulnerabilities in both software and hardware domains. The research by Ferrag *et al.* [14] shows that LLMs can easily read and analyze source code to discover security bugs and vulnerabilities. Recent works find that LLMs can show competitiveness to traditional methods in terms of vulnerability detection tasks.

For example, Chen *et al.* introduced "DiverseVul", [37] is a vulnerable source code dataset that covers a wide range of programming languages. When applying this to security, their research demonstrated that LLMs trained on security-related datasets are able to discover complex vulnerabilities that may be difficult for classical linear static analysis tools to identify on their own, which is crucial for detecting those more advanced security issues that traditional rule-based systems have trouble detecting. In addition, "LineVul", a line-level transformer model initiated by Fu *et al.* [38] predicts software vulnerabilities. The model was evaluated on an extensive dataset of C/C++ functions. It was demonstrated to be highly effective at identifying vulnerable code with respect to the most dangerous 25 Common Weakness Enumerations (CWEs).

2.2.2 Malware Detection and Classification

LLMs have demonstrated great potential for malware detection and classification tasks. They can even understand the structural pattern of a code, Application Programming Interface (API) calls, and natural language description of its behavior to make a classification of malicious software. Demirkıran *et al.* [39] showed that transformer-based models could achieve good performance in classifying malware family based on a sequence of API-call as features. Their findings demonstrated that these models outperformed more classical machine learning approaches in the F1-score as well as AUC scores. This work obtained the best performance on several benchmark datasets, demonstrating the promising abilities of LLMs in malware analysis tasks. Other work by Joyce *et al.* introduced "AVScan2Vec" [40] that maps antivirus scan reports into vector representations. It works well with any task, such as the classification of malware, clustering analysis, and also in nearest neighbour search. These studies show that LLMs can be used at scale to parse and interpret malware information.

2.2.3 Network Intrusion Detection

Network intrusion detection is another area of application for LLMs that has changed the way security systems detect and deal with potential threats. The study of Wu *et al.* [41] introduced Robust Transformer-based Intrusion Detection System (RTID). The system achieved better performance than traditional machine learning and deep learning techniques on CICID2017 and CIC-DDoS2019 datasets. They demonstrated the applicability of LLMs to allow for improved security analysis of complex network traffic data. This trend was further confirmed by Ferrag *et al.* [14], which provided evidence on how LLMs are efficient in:

- Process and analyze network traffic patterns
- Identify anomalous behavior in real-time
- Better performance and reduced false positive rates compared to traditional methods
- Adapt to new types of attacks through continuous learning

2.2.4 Phishing Detection

Phishing detection is another area where LLMs have shown significant promise. These models can analyze email content, URLs, and other metadata to identify potential phishing attempts. The application of LLMs in phishing detection has shown remarkable progress in identifying and preventing social engineering attacks. Koide *et al.* [42] presented ChatSpamDetector, which uses LLMs to effectively detect phishing emails with an impressive 99.70% accuracy. This system not only flags phishing attempts but also explains why it marked them as a result, users can learn about the choices of potentially suspicious emails. Another research by Jamal and Wimmer [43] who developed IPSDM, an improved model based on the BERT family. It was specifically fine-tuned to detect phishing and spam emails. Their research showed that IPSDM had excellent accuracy, precision, recall, and F1-score results on both balanced and unbalanced datasets.

2.2.5 Automated Threat Intelligence

LLMs are increasingly being used to automate the process of gathering, analyzing, and disseminating threat intelligence. They have greatly improved how threat intelligence is gathered and analyzed. LLMs can handle large amounts of security data and turn it into useful insights. This ability has changed how organizations spot and react to new security threats [14]. Evange *et al.* [44] found that transformer-based methods perform better than previous top approaches for Named Entity Recognition (NER) in threat intelligence. Their study used the dataset for NER in Threat Intelligence (DNRTI), which includes over 300 threat intelligence reports that contain 175,220 annotated words in 13 classes. This research showed that LLMs are effective at identifying cybersecurity-related entities.

2.2.6 Hardware Security

One of the latest applications of LLMs is in hardware security. Although this is a new field of study, early research has been very promising. Fu *et al.* [5] introduced "LLM4SECHW" recently, it is a proposed framework for hardware debugging based on a domain-specific Large Language Model (LLM) fine-tuned on a custom dataset. Their research demonstrated the effectiveness of LLMs in the following areas, such as:

- Locating bugs in hardware design
- Giving suggestions on debugging
- Road to automated quality control in hardware design

Additionally, Ahmad et al. [45] investigated the feasibility of the LLMs on hardware security bugs. Their research demonstrated that models such as GPT-3 and CodeGen can successfully synthesize replacement code that addresses security vulnerabilities in hardware designs. It can also automate hardware bug repair experiments yielding promising results, with success rates varying according to the type of bug and the LLM model used.

2.3 Challenges and Limitations in Applying LLMs to Cybersecurity Tasks

Although Large Language Models demonstrate great potential for several cybersecurity uses, it is not easy to implement. This section presents the major limitations and challenges when using LLMs in cybersecurity tasks.

2.3.1 Model Size and Deployment Challenge

The first and probably the most critical challenge of using LLMs in cybersecurity is their sheer size and computational demands. They have become much bigger over time, reaching 117 million parameters for GPT-1 and going up to 175 billion for GPT-3 [46] [7]. The scaling drawbacks are significant, such as:

- Storage requirements
- Memory management
- Real-time processing capabilities
- Deployment costs in resource-constrained environments

Training and deploying these models need a lot of computational power. For example, training a model like GPT-NeoX-20B requires many high-end GPUs. It was trained on twelve Supermicro AS-4124GO-NART servers, each equipped with eight NVIDIA A100-SXM4-40GB GPUs [47]. These demands make it hard for many organizations and researchers in cybersecurity to access and use such models.

2.3.2 Computing Power and Resources

The computational demands of LLMs present a significant barrier to their widespread adoption in cybersecurity applications. As highlighted by Xu *et al.* [7], running advanced LLMs requires:

- High-performance GPUs or TPUs
- Substantial memory resources
- Special infrastructure for deployment
- High energy consumption

These requirements lead to high operational costs, especially for real-time security applications. Running large LLMs also uses a lot of energy, raising concerns about environmental impact and sustainability [14]. Nowadays, the environmental impact of LLM deployment has emerged as a critical concern. Studies by Patterson *et al.* [48] indicate that training LLMs can use as much energy as several hundred households do in a year. This high energy use is especially concerning because these models also need frequent updates to keep up with new security threats. Another research by Strubell *et al.* [49] found that training one large transformer model can emit approximately 626,155 pounds (284 metric tons) of CO_2 equivalent, which is nearly five times the lifetime emissions of an average American car. This environmental impact raises serious questions about sustainability and responsible LLM use in cybersecurity.

2.3.3 Data Scarcity and Quality

Another challenge is the lack of high-quality domain-specific data to train LLMs in cybersecurity. Zhang *et al.* [50] addressed this issue in their review that many cybersecurity datasets are limited in size and diversity. Such approaches often lead to models that perform well on benchmark tests but poorly against actual cybersecurity problems. Also, the quality and representativeness of the data at hand matter a lot. Researchers identified the risk of dataset contamination due to repeated filtering of the same data, which created an overlap between training and testing sets. This can make performance metrics look better than they are. The challenges related to data include [7]:

- Limited availability of labelled security datasets.
- Difficulty in obtaining real-world attack data because no one wants to share critical data.
- Imbalanced datasets that do not represent all attack vectors
- Lack of quality verification of training data

2.3.4 Privacy and Security Concerns

The use of proprietary LLMs in cybersecurity introduces its own set of privacy and security concerns. It raises significant privacy and security concerns that extend beyond traditional cybersecurity challenges. The integration of LLMs into security systems introduces new attack risks and vulnerabilities that must be carefully considered [16] [51]. One major concern is the exposure of sensitive information during the training process. LLMs might accidentally memorize confidential data from their training sets. A study by Carlini *et al.* [52] has demonstrated that LLMs can unintentionally memorize and expose sensitive information from their training data, including personally identifiable information, API keys, and security credentials. This is particularly problematic in the world of cybersecurity, where models can be dealing with very sensitive data. Fine-tuning models on organization-specific security information, in particular, increases the likelihood of data exposure through leakage. In addition, Xu *et al.* [7] brings concerns regarding model tampering and poisoning attacks. Threat actors can manipulate the model through training data or create adversarial examples to create holes in the model architecture to influence the security decisions it makes. The use of LLMs for automated decision-making, particularly in such critical security applications, can pose quite serious risks.

2.3.5 Interpretability and Trustworthiness

The black-box nature of LLMs is one of the major challenges in building trust and ensuring interpretability in cybersecurity. As noted by Liu *et al.* [53], the complexity of these models makes it hard to understand how they make any decisions. This is a critical issue in cybersecurity, where false positives or negatives can have serious consequences. The lack of interpretability also makes it difficult for security professionals to validate the model's decisions and ensure they comply with security policies and regulations. A separate study from Yang *et al.* [54] also emphasizes the trustworthiness that is needed when LLMs are employed in sensitive environments and put into critical infrastructure. The opacity around how these models make decisions can lead to reluctance towards adopting them for critical security activities. Additionally, this problem is made worse by model hallucination, in which the generated suggestions by LLMs appear valid but are not [55]. Implemented uncritically, it could blow a hole in security.

2.3.6 Adversarial Attacks and Model Robustness

LLMs are vulnerable to adversarial attacks, which represents a significant concern in cybersecurity applications. LLMs are vulnerable to different types of adversarial. As LLM technology has advanced, so have these types of attacks, creating a constant challenge for security professionals [50]. Prompt injection attacks have become a serious threat. Research by Liu *et al.* [53] demonstrated that carefully crafted prompts can trick LLMs into producing malicious outputs or bypassing security measures. This can be very dangerous in security-critical applications where the model's output directly affects security decisions. The researchers also discovered that even advanced models could be vulnerable to subtle manipulations that take advantage of their learned patterns and behaviours. Moreover, Ferrag *et al.* [14] outlined some of the most serious vulnerabilities of LLM-based security systems, such as prompt injection, insecure output handling, and training data poisoning. These vulnerabilities can cause models to behave in ways that stray from intended security protocols. The researchers emphasized that traditional security measures might not be enough to defend against these new types of attacks, highlighting the need for specialized defenses for LLM-based security systems.

2.3.7 Domain Adaptation and Generalization

LLMs work well for a wide variety of natural language processing tasks, but adapting them to a specific cybersecurity domain remains challenging. Because cybersecurity threats are complex and evolving rapidly, these models must also respond to new patterns and contexts. However, domain adaptation and generalization are some of the most technical challenges of LLM-based security systems. LLMs that have been trained on general-purpose data will not adapt well to specialized cybersecurity domains. This limitation is especially visible when models face unseen attack patterns or highly technical security scenarios that strongly differ from their training set [5]. Research by Abdali *et al.* [56] observed that in the face of novel security threats that they never trained on, LLM performance could reduce dramatically. This failure to generalize poses serious risks in cybersecurity implementation settings, where novel, unanticipated, and never-before-seen attack vectors are constantly emerging. The researchers write that while LLMs show an ability to handle known security patterns, their capability to generalize to new threats is still limited.

Also, many of the cybersecurity tasks are so specialized that models need to grasp highly technical detail and domain terms. Xu *et al.* [7] note the need for detailed fine-tuning or domain expertise training, which is a challenge since high-quality domain-specific data are scarce. The researchers note that successful adaptations of their model to cybersecurity domains will require both technical modifications of the model itself and a thorough understanding of the particular security context in which the model will be employed.

2.4 Emerging Trends and Future Directions in LLM-based Cybersecurity Solutions

LLMs in cybersecurity are evolving quickly. Trends and developments are influencing new usages of these models in security applications. This progress plays a role in solving today's issues but also opens up new horizons for improved security and threat detection.

2.4.1 Domain-Specific LLMs

The recent development of LLM has focused on the aspect of fine-tuning the models and creating domain-specific custom LLM to enhance the model's learning of the task at hand. A recent development is that there are researchers working on building and optimizing LLMs for security-specific workloads. For instance, Fu *et al.* [5] developed LLM4SECHW, and Ferrag *et al.* [10] proposed a privacy-preserving lightweight BERT-based architecture for the detection of IoT threats. The two teams utilized medium-sized LLMs and customized them with a fine-tuning process containing their own datasets. Their results are impressive as both the fine-tuned models outperform general LLMs that are trained on general datasets. Their work shows the need to design model architectures and training procedures specialized to the various cybersecurity domains for more efficient and accurate results.

2.4.2 Efficient, Lightweight and Multimodal LLMs

Resources are limited, and training large models can be very expensive, so the efficient and lite LLM architecture building is another important trend in the field. A literature review study from Zhang et *al.* [50] showed that high performance could be maintained with significant reductions in computational requirements by careful architecture design and applying model-optimization techniques.

The use of LLMs which integrate several forms of data can lead to exciting new use cases for cybersecurity, these multimodal approaches enable LLMs to handle and analyze different kinds of security-related data at the same time. In the current cybersecurity landscape, where threats traverse numerous data types and formats, this capability can be incredibly valuable. Xu et al. [7] have found in their literature review that LLMs could process multimodal data sources (even joint complex network traffic patterns) such as JSON log data and network packet binary code. Which in turn scaled the number of false positives and the detection accuracy of the indicator across diverse data types. It improves the efficiency and reliability of threat detection systems.

2.4.3 Explainable AI in LLM-based Cybersecurity

Critical cybersecurity systems need transparency and interpretability in the security-critical decision-making process. As a result, developing explainable AI capabilities in LLM-based cybersecurity solutions has emerged as a new research direction. The integration of explainable AI techniques with LLMs allows security specialists to understand the reasoning behind model decisions, which is crucial for validating alerts and maintaining regulatory compliance [16]. For example, Ahmad *et al.* [57] proposed FLAG (Finding Line Anomalies in Code with Generative AI), the first attention-based mechanism for bug detection. The model gives clear explanations of the vulnerabilities, which makes it convenient for security analysts to analyze the justification of the model's decisions.

2.4.4 Adversarial Training and Robustness

As the variety and nature of cyberthreats are complex and continuously evolving, researchers have since shifted gears to developing an all-around LLM resistant to all forms of adversarial attacks. Yao *et al.* [16] reflected this need in their survey on LLM security and privacy. In their research, they discussed important security challenges and proposed solutions by suggesting a framework for LLM-based security systems. It also highlighted the need to combine diverse defensive approaches, including adversarial training, input validation, and continuous model monitoring, to make a robust system. Another interesting example comes from Deng *et al.* [58] introduced the "MASTERKEY" framework, which highlights the serious vulnerabilities of LLM chatbots to adversarial inputs. The study emphasized the importance of adversarial training, input validation, and ongoing model monitoring to improve the robustness of LLMs.

2.5 Research Gaps and Future Directions

LLMs in cybersecurity bring many opportunities in order to create more robust LLM-powered solutions these research gaps need to be filled. For example, focusing on open-source LLM-based solutions will give organizations more control over the system and will allow better customization to their needs.

2.5.1 Current Limitations in Literature

The current research landscape reveals several significant limitations that need attention from the research community. A primary concern is the predominant focus on proprietary LLM solutions, which makes it impossible for organizations to adopt and customize them with full control. Although models such as GPT-4 and Claude have shown impressive capabilities, they are closed-source and unsuitable for sensitive security environments, in which organizations require tools that they can fully control—something that closed-source models cannot offer.

Data Representation and Quality: One of the major limitations found in existing research is the inability to represent most real-life cybersecurity scenarios in training data. Xu *et al.* [7] point out that most of the existing studies are based on syntactic or really old data sets that may no longer represent current threat landscapes. This gap is even sharper in the area of hardware security applications where Fu *et al.* [5] stated that because there are no high-quality, labelled hardware vulnerability datasets, it is really difficult to develop effective LLM-based detection systems.

Privacy and Data Control: For organizations that deal with sensitive data around security, public proprietary LLMs like Gpt-4 present a major risk. Yao *et al.* [16] pointed out that few articles in the literature conducted in-depth studies on the negative effects of deploying proprietary LLMs in security-critical scenarios, especially the problems caused by data privacy and control. The way these companies are collecting and handling data is still not clear. For instance, the recent incident where data privacy concerns led Samsung to ban the use of ChatGPT highlights this gap in present-day research and implementation approaches [59].

Evaluation Metrics and Benchmarks: One of the major limitations is the absence of standardized evaluation frameworks to evaluate LLMs for cyber tasks. Different metrics and benchmarks are used in the current literature, preventing meaningful comparisons of different approaches. The inconsistency in evaluation methodology not only makes the comparison of the efficacy of different security LLM architectures and approaches challenging but also raises concerns about the real effectiveness achieved from the different security applications of LLM.

Integration Challenges: Studies focused on the pragmatic incorporation of LLMs into current security frameworks are scarce. Theoretical frameworks are different from proof-of-concept implementations, Ferrag *et al.* [14] highlighted difficulties in deploying systems in practice such as latency on systems, resource optimization, and integration with legacy systems are surprisingly limited.

Resource Requirements and Scalability: While many open-source models like LLama and Falcon recently emerged, there has not yet been much research into optimizing these models for resource-constrained security environments. Fu *et al.* [5] also stated that although there is some interest in applying this technique to real-world security tasks, most studies have focused on performance metrics at the expense of practical deployment metrics like resource usage and scalability of performances in real security contexts.

2.5.2 Areas for Further Research

In order to bridge these gaps and progress in the field of cybersecurity, the following key research should be prioritised:

- *Develop Advanced Model Architectures:* The research agenda should focus on the creation, open-sourcing, and security-oriented purpose-design of high-performance models. The performance of specialized architectures could be drastically lower on computational requirements while not losing efficiency on security tasks, already a few studies have shown this.

- *Standardization of Evaluation Frameworks:* There is a need for a unified evaluation framework for security-oriented LLMs. This will enable organizations to run their proposed system over consistent benchmarks that would allow for much more meaningful comparisons of individual personal solutions.

- *Enterprise Integration Strategies:* More work is needed on creating complete solutions for integrating open-source LLMs in enterprise security ecosystems It is also researching scalable methods for deployment, frameworks for monitoring, and integration with existing security tools.

3 Methodology

3.1 System Design

The goal of this research is to design a system and test its performance for network traffic analysis and cyber incident reconstruction. The core architecture is built upon locally run LLMs. This system utilizes the LLMs and is capable of providing insightful information regarding the network traffic data. It addresses the major issues of traditional methodologies and works as a helpful personal assistant in analyzing large amounts of data. It also provides a comprehensive framework to create an effective network traffic analyzer that is context-aware and provides intelligent responses for security analysis. The system design criteria can be listed as:

1. Develop a chatbot for processing and analyzing network traffic data (PCAP files).

2. Collect threat intelligence data and integrate that information for providing enhanced security analysis.

3. Implement a database system for efficient storage and retrieval of relevant information so that LLMs can provide insightful and context-relevant responses.

4. Develop and test a framework that can be used for processing and analyzing large amounts of network traffic data (PCAP file) at the same time maintaining high accuracy in threat detection.

These design criteria align well with the growing concern and the need for sophisticated cybersecurity analysis tools as highlighted by many researchers [10] [7] [5]. It also demonstrates the possibilities of security analysis tools that use LLMs as their core processing mechanism.

3.2 Hypotheses and Expected Outcomes

Based on this thesis's literature review and work, this thesis sets several hypotheses to address the limitations and challenges of modern cybersecurity analysis and incident reconstruction. Most of these hypotheses are identified based on future trends, which were analyzed in Chapter 2, and the design criteria outlined in the previous section. This study proposes the following hypothesis:

- H1: Integrating Locally run LLMs will improve the speed and accuracy of network traffic analysis for cyber incident reconstruction. At the same time, it will ensure the privacy of data.

- H2: Getting data from VirusTotal API and feeding that information to the LLM will provide more background context and required threat intelligence. As a result, it will provide more comprehensive threat intelligence for cyber incident reconstruction.

- H3: Storing custom data in a vector database will enhance the LLM's ability to provide contextually relevant responses to that specific custom data. As a result, more accurate threat identification will be possible.

- H4: Developing an LLM-powered private chatbot will allow users to ask questions and get informative instruction or assistance, which will make their task a lot easier.

The first hypothesis (H1) points out that, integrating locally run open-sourced LLMs for network traffic analysis will significantly increase the speed and accuracy of the cyber incident reconstruction process. This hypothesis addresses not only performances but also crucial data privacy concerns. Local LLM deployment allows the user to process sensitive network data without compromising privacy while harnessing the advanced analytical capabilities of these models.

The second hypothesis (H2) addresses the quality and comprehensiveness of threat intelligence for cyber incident reconstruction and points out that it will be increased due to the integration of VirusTotal API data. It will boost LLM's knowledge base with real-time threat intelligence from VirusTotal. As a result, the system is expected to provide more detailed and contextually relevant analysis.

The third hypothesis (H3) proposes that storing custom security data in a vector database will increase the performance of the LLM by providing responses that are more contextual. This hypothesis is based on the vector database's ability to maintain semantic relationships across multiple data types, as a result, it prduce more accurate threat detection. This hypothesis is also supported by the recent work of Zhao *et al.* [55], who show how retrieval-augmented generation mechanisms can improve LLM capabilities and performance across specialized domains.

The final hypothesis (H4), which is the user interaction component of the system, proposes that an LLM-powered private chatbot will serve to increase both the accessibility and utility of security analysis to end users. This acknowledges the increasing impact of intuitive interfaces in security tools and suggests that natural language interaction could improve security operations. The chatbot is likely to respond clearly to each security query in a context that will ensure security analysis for security teams with different skill levels to understand even the most complex issues. This will help to improve how security analyses are explained and ensure everyone can follow along. Testing these hypotheses would lead to the following expected results:

This study anticipates demonstrating a quantifiable improvement in the efficiency of security incident analysis by developing a solution that combines vector databases with local LLMs. This solution's effectiveness will be evaluated using precision, recall, and F1 scores as baselines, particularly for complex attack patterns and zero-day threats. Secondly, the work intends to create a paradigm shift in context-aware security analysis using LLMs with vector database integration, enabling the transfer of contextual semantic understanding. This is a critical gap addressed in recent studies since almost all current security analysis tools are context-impaired. Thirdly, this work expects that this domain-specific framework will produce more concrete and comprehensible security solutions using human-interpretable natural language analysis. This outcome aligns with the growing need for more readily available security intelligence in contemporary organizational settings — which is also reflected in the recent academic literature.

3.3 System Workflow

The framework of the proposed system architecture combines traditional packet capture analysis with modern LLM-driven techniques. The design is essentially a pipeline that converts raw network data into actionable security insights via interrelated components. This design fits well with the trends of cybersecurity automation and effectiveness, as noted by many researchers [10] [55] [5], highlighting the immense power of combining large language models into a security architecture.

3.3.1 Role of Major Components

The system architecture consists of four key modules that lie at the center of the security analysis pipeline of this workflow:

PCAP Analysis Module: It acts as the first intake point of data, taking network packet captures in their raw state and extracting the security-related features from them. It performs a detailed examination of packets to detect potential security breaches and unusual traffic activities.

VirusTotal API Integration: The system uses an external threat intelligence source, VirusTotal API, to give an enriched analysis with a substantial amount of malware and threat data. This integration provides multi-engine analysis and threat intelligence that extends the internal analysis functionality of the system.

Vector Database: Acting as a centralized knowledge repository. The vector database stores and manages high-dimensional representations of security-related data. It enables efficient similarity searches and contextual retrieval, maintaining historical context and providing quick access to information during analysis.

Large Language Models: The system implements locally run LLMs to provide advanced natural language understanding and generation capabilities. These models interact with the vector database to supply contextualized security insights and suggestions. It is the heart of the systems architecture that uses advanced reasoning capabilities to solve and provide insightful guidelines.

3.4 Vector Database

Vector databases have become the foundation of modern information retrieval for LLM systems and Generative AI. The primary goal of combining a vector database into the system is to store vector data in a structured manner, allowing for semantic search and semantically evaluating relevant information based on query content.

3.4.1 Purpose of Vector Database in the System

The vector database performs several important functions in this architecture. Its primary function is to act as an intelligent repository, preserving the semantic relationships between various types of security events, network patterns, and threat indicators. This ability is especially useful in cybersecurity, where the context of each event does not always match an actual cyber incident; the relationship between two independent events can be useful in both threat detection and incident reconstruction. Furthermore, similarity search in vector databases improves performance, which is computationally expensive in traditional database systems. To optimize data storage and retrieval, the Vector Database stores embeddings, which represent input data as dense vectors. This is the foundation of semantic embeddings, which identify the meaning of data rather than relying on keywords. It makes it possible to conduct more intelligent searches, including identifying connections and relationships that might not be clearly indicated in the raw data. The vector database is incorporated into this system architecture primarily for the following reasons:

Efficient Similarity Search: Vector databases support similarity search, allowing for quick searches over large amounts of network traffic data, which can help detect patterns and anomalies in near real-time [55]. This ability is especially important when it comes to identifying potential security threats that might be similar to known malicious patterns.

Dimensional Reduction: The vector database allows efficient indexing and storage of the high-dimensional nature of network traffic data [60]. This approach gives a significant advantage in avoiding the computational overhead for larger network traffic with big packets.

Scalable Data Management: The vector database architecture provides a scalable solution to handle growing volumes of network traffic data while maintaining quick response times for queries.

Since the vector database reduces the search space map, it helps the LLM get more relevant information for generating responses, which causes it to play a significant role in system performance and scalability. For larger datasets, searching throughout the entire database can be computationally and time-consuming.

3.4.2 Embedding Creation for Similarity Search

This embedding process serves as a link between raw network traffic data and vector databases. The methodology begins with analyzing PCAP files using a Python script, which generates preliminary results that serve as the foundation for the creation of the embedding shown in Figure 3.1 as part of the system design shown below.

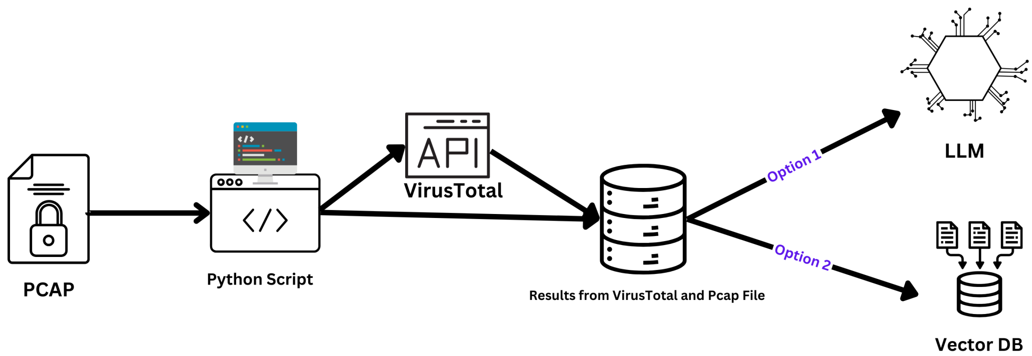

Figure 3.1: Partial System Diagram Showing the Data Preprocessing and Feature Extraction

This system converts PCAP file output into high-dimensional vectors using an embedding model to capture the most relevant network traffic patterns. This conversion is required for similarity search and pattern recognition. According to a study, Ferrag *et al.* [10] found that the strength of the embeddings has a significant impact on overall performance, particularly in terms of detecting potential disasters and anomalies. This system utilizes only locally running models to produce the embeddings. The system processes network traffic data using two different approaches, as shown in the architectural diagram shown in Figure 3.1 created for this project

(Options 1 and 2).

In option 1, the system sends pre-processed data to the specialized LLM for direct analysis. Based on the user's query, LLM analyzes the PCAP file using Python script (TShark command) and returns answers. In contrast, option 2 generates embeddings and stores them in a vector database before sending them to the specialized LLM. As a result, there are two distinct approaches to real-time market analysis and trend discovery.

This vector database infrastructure utilizes these embeddings to enable immediate retrieval of similar traffic patterns and security-related pieces of information using a similarity search mechanism based on the user's query. Using structural and semantic-level features of the network data with advanced metrics for computing similarity, the system permits complex pattern matching for detection beyond simple signature-based types of detection methods.

3.4.3 Query Handling Using Vector DB

After the embeddings are generated and saved in the vector database, they can be retrieved by comparing the vector expressions of the query. This process is referred to as a similarity search, which essentially calculates the cosine similarity or other distance metrics with respect to the query vector and the document vectors [61].

The vector database returns a ranked list of n most similar documents according to these computed similarities, which is then fed to the LLM as context to help it generate a relevant response. This drastically limits the information that the LLM needs to deal with and allows it to answer quicker and with more precision. Figure 3.2 shows a typical flow of handling queries using a vector database, where the query is embedded into a vector space, and then compared in similarity to all stored document vectors, after which the document with the highest similarity score is retrieved.

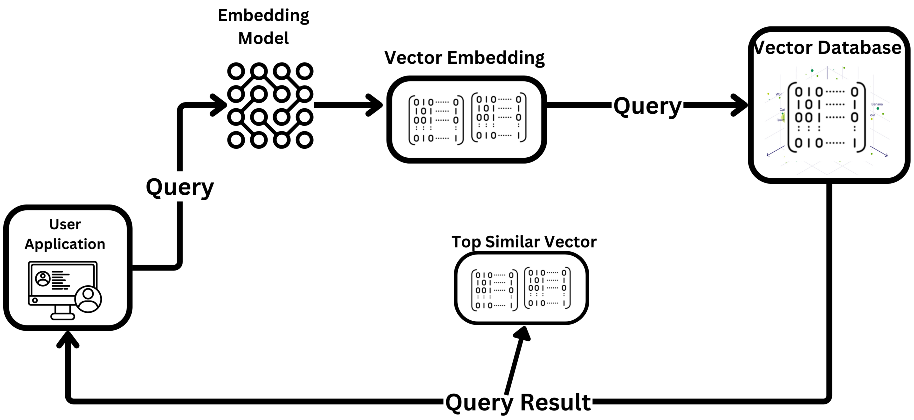

Figure 3.2: Vector Search

3.4.4 Query Processing Pipeline

This system workflow follows a query processing structure where processing takes place in a sequential manner with the vector database tech and the LLMs providing power respectively. This starts from query transformation and goes up until response generation. The process of handling the query is presented in the Figure 3.1 diagram where the query follows a path through several layers of processing. The query-handling engine is built around the interaction of the embedding model and the vector database. This traces the embedded model, which takes a query input in natural language and converts it to a vector representation to fit the database structure. Query processing is performed in a 5-phase pipeline:

- *Stage 1 - Query Embedding Transformation*: First, the system will convert the incoming query into a vector using its embedding model (the same one that was used at the time of the initial data ingestion step). This guarantees that there is uniformity between the vectors we query and the vectors we store, which is an important aspect of similarity matching accuracy.
- *Stage 2 - Vector Database Search*: Then, the transformed vector is used as a query and perform a similarity search within the vector database. This involves searching over-optimized indexing structures that enable the fastest

search for the most relevant matches based on vector similarity metrics.

- *Stage 3 - Context Retrieval*: The system retrieves applicable contexts based on the vector similarity search results. These contexts include linked metadata and security-relevant information that provide extra depth to the query response.

- *Stage 4 - LLM Integration*: These retrieved contexts are then forwarded to the LLM component of the system to be processed along with the original query. The architecture diagram of this system also shows how it uses only locally run LLMS for natural language understanding and response generation integration.

- *Stage 5 - Response Generation*: Lastly, it generates an elaborated answer, taking advantage of the accuracy of vector similarity-based relevance matched to the domain context solutions.

3.5 LLM Utilization

In this proposed system architecture, locally run LLMs play the main role in understanding user queries, generating human-like responses, and interacting with the vector database for efficient information retrieval.

3.5.1 Selection of LLM Models

The thesis work employs only open-source models that can be run on local devices, reducing concerns about security, privacy, and local control. This model also addresses the very serious security risks associated with public commercial LLMs, ensuring complete data authority. This system, in particular, uses a specific set of Llama 3, Llama 3.1, Falcon, and Mistral models with sizes ranging from 8 to 32 billion parameters, which are easily runnable on dual NVIDIA TITAN RTX-based hardware infrastructure with 24GB of VRAM deployed in the system hardware.

The choice was made not only according to performance needs but also practical limitations. The applicability of LLM for security-related purposes is quite specific to the model's architecture, scale, and training, as noted by Xu *et al.* [7]. Larger models such as GPT-4 have shown generally good performance, but this work shows that with this system framework, locally run models can do much better and match performance on specific security-analysis tasks while providing privacy of data.

This work was performed using NVIDIA TITAN RTXs with 24576MiB memory, ensuring capability for fast model inference while acting as a realistic constraint on model size. This configuration provided both optimization and balance between model performance and resource utilization. This setup offered optimization and a balance of performance versus resource use. Models of 8-32 billion parameters are selected to balance processing speed and security analysis accuracy and pick optimal model parameters that provide the best performance for specific tasks. Recent research in the field also supports the choice of these particular models.

The Llama model family, for example, was reported by many researchers to be surprisingly efficient on narrow tasks. Likewise, Falcon and Mistral models have performed well in security use cases, especially when fine-tuned within well-defined domains. This implementation takes advantage of these properties while allowing full control over the inference and data flows.

4 System Design and Implementation

4.1 System Overview

The proposed system aims to use the power of Large Language Models and vector databases and provide context-aware querying and generative AI responses for network traffic analysis. The system aims to help security analysts in catching and investigating potential security threats. It does this by processing packet capture (PCAP) files, integrating external threat intelligence sources, and offering intelligence-based querying capabilities.

4.1.1 Problem Scope and Use Case

As the volume and complexity of enterprise network traffic continue to grow, human analysis has become impractical and inefficient. The problem for security analysts is that they need to rapidly assess most threats and then respond quickly to remediate them. The proposed answer to this problem is to utilize the power of LLMs and present a framework that automates feature extraction from PCAP files, complements them with a threat intelligence database, and enables LLM-powered querying and response generation. This use case falls into the category of security analysts being required to conduct an investigation into a potential security event, such as malware infection, phishing attack, or data exfiltration. Analysts can use the system's context-aware querying and generative AI replies to find associated events, detect patterns, and learn about the nature and extent of occurrences.

4.1.2 High-Level Architecture Description

As shown in the full system architecture in Figure 4.1, the proposed system comprises four high-level architectural components:

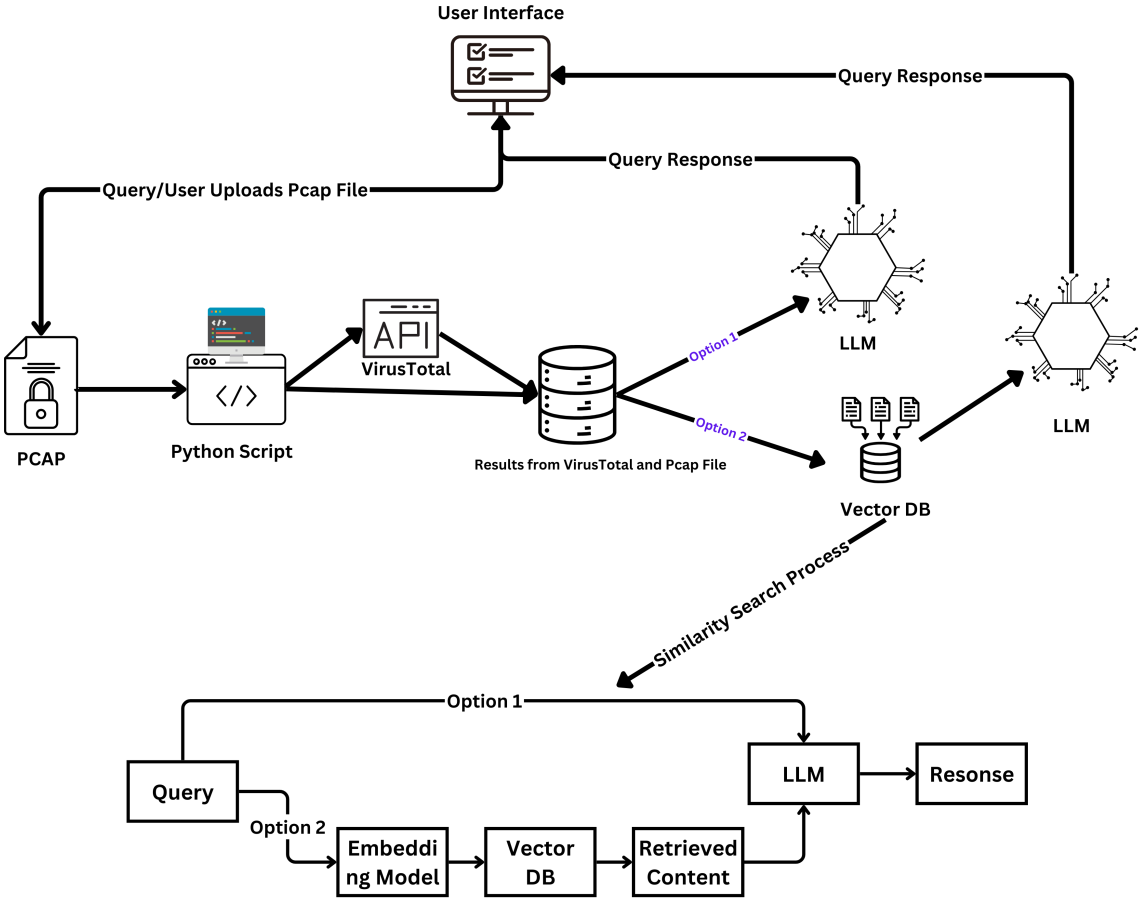

Figure 4.1: System Design Architecture

1. PCAP File Preprocessing Pipeline - this takes care of ingesting and parsing PCAP files, extracting features and also integrating external threat intelligence feeds like VirusTotal.

2. Store in Vector Database - The part of the system that deals with generating vector embeddings for preprocessed network traffic and storing it in a vector database, which is used to search (similarity search) retrieved data.

3. LLM Query Processing and Response generation - This component works with LLMs running locally to get the user query, search for related context in the vector database and generate AI responses.

4. User Interface - A user-facing part of the system that allows security analysts to communicate with it by sending queries and getting responses.

The data preprocessing pipeline converts raw PCAP files into structured JSON format, which can then be evaluated and integrated with the vector database and LLM agent helpers. This consists of three major steps: reading PCAPs and extracting features, integrating with the VirusTotal API, and converting to JSON file. The vector database integration component produces vector embeddings of the preprocessed network traffic JSON. It allows fast similarity searches and retrievals utilizing ChromaDB [62], a high-performance vector repository intended for efficient similarity matching and retrieval. The LLM query processing and response generation component uses locally hosted language models to understand inquiries, retrieve relevant material from the vector database, and synthesize LLM-powered responses. The LangChain [63] library is used to integrate OLLAMA models with the vector database and manage the question handling and response fabrication workflow.

4.2 Technical Implementation

4.2.1 PCAP Parsing and Feature Extraction

The first step of the data preprocessing pipeline is parsing the raw PCAP files and extracting meaningful features appropriate for analysis. In this stage, PCAP files are read using PyShark [64], which is a Python wrapper for the TShark library from Wireshark, and features are extracted, including IPs, port numbers, protocols, and payload data. This system converts raw network capture data into embeddings suitable for analysis through a multi-stage pipeline. In the last step, TShark (terminal-based network protocol analyzer) is used to convert PCAP file data into structured JSON format. Compared to working at a packet level, it has multiple benefits, such as better data structure and more efficient processing. The information transformation method follows a systematic workflow.

Listing 4.1: Libraries Used for PCAP Preprocessing Pipeline

```
from langchain_community.llms.ollama import Ollama
from langchain.document_loaders import JSONLoader
from langchain.text_splitter import
   RecursiveCharacterTextSplitter
from langchain_community.embeddings import OllamaEmbeddings
```

This implementation uses JSONLoader to parse the structured data from the network and then the RecursiveCharacterTextSplitter for optimal content segmentation. This method is consistent with the research conducted by Xu *et al.* [7], which emphasizes the significance of appropriate data segmentation in security analysis systems. The transformation pipeline is represented as follows:

$$\texttt{PCAP} \rightarrow \texttt{JSON} \rightarrow \texttt{Text Segments} \rightarrow \texttt{Embeddings} \rightarrow \texttt{Vector Storage}$$

This sequential transformation ensures network traffic data is properly structured and contextualized before embedding generation. A study by Ferrag *et al.* [10] emphasizes the significance of appropriate data segmentation in security analysis systems.

4.2.1.1 VirusTotal API Integration

The system uses external threat intelligence data sources to improve the collected cyber threat features. It compares IP addresses, domain names, and file hashes to the VirusTotal database [65] through API calls to see if they are associated with any known malicious activities. These API queries return information about each distinct attribute parsed from network traffic. After that, the characteristics and threat data are combined for further processing.

4.2.1.2 JSON Conversion

The final stage of the data preprocessing pipeline is to transform both the integrated feature and the integrated threat intelligence information into JSON format. JSON (JavaScript Object Notation) is a lightweight data transport format that is both human-readable and machine-parseable [66]. Changing the data to JSON allows for better interaction with vector databases and LLMs, as well as smooth integration with other tools/systems. The JSON Conversion function converts features and threat intelligence data into a hierarchical arrangement, in which each network traffic record is saved as a JSON object with varied characteristics in key-value pairs. This JSON data is subsequently saved to a file or passed straight to the system's next component.

4.2.2 Embedding Generation and Vector Storage

This system uses "OllamaEmbeddings" to generate vector representations of the processed network data. The decision to use this library was driven by several factors, including its local processing capabilities and seamless integration with our chosen LLM implementation. Ollama is a platform that provides tools for running and interacting with large language models locally, including the capability to generate embeddings. In this project, the' nomic-embed-text' model was used for embedding tasks. This model is specifically designed to convert data into dense vector representations and effectively capture the semantic meaning and relationships between different features. The embedding generation process is as follows:

Listing 4.2: Embedding Creation Process Code

```
from langchain_community.embeddings.ollama import
   OllamaEmbeddings

base_url = "http://localhost:11434"  # Update IP and port as
   needed

def get_embedding_function():
    """Get embedding function using Ollama"""
    try:
        embeddings = OllamaEmbeddings(
            base_url=base_url,
            model="nomic-embed-text"
        )
        return embeddings
    except Exception as e:
        print(f"Error␣initializing␣embeddings:␣{e}")
        raise
```

The generated embeddings are saved within ChromaDB, a vector storage solution. The integration between embedding creation and storing is intended to maintain semantic relationships while allowing efficient retrieval. Embedding generation involves the following steps:

- The preprocessed JSON data is loaded into memory using the JSONLoader utility, which is specifically designed to handle structured JSON data efficiently.

- To efficiently handle huge datasets, split JSON data into smaller chunks using the "RecursiveCharacterTextSplitter".
- Applying the OLLAMA Embeddings model to each chunk of JSON data to generate the corresponding vector embeddings.
- Store the generated vector embeddings in the ChromaDB vector database for a more optimal and efficient search and retrieval of similar vectors.

4.2.2.1 Similarity Search and Retrieval

ChromaDB helps perform fast and efficient similarity searches and user queries based on the retrieval of relevant network traffic data. Once a user submits a query, the system first creates a vector embedding of the inputted query using the same OLLAMA Embeddings model used on the preprocessed data. Next, the query vector is compared within the locally created ChromaDB database using a similarity metric, such as cosine similarity. ChromaDB computes similarity scores and returns a ranked list of the closest vector embeddings from the set. These top-k (similarity) results are then used to fetch the respective preprocessed data from JSON in order to provide context to the LLMs so they can generate accurate and relevant responses back to the user query.

[Figure 5.4: Similarity search and retrieval process using ChromaDB.]

4.2.3 Local LLM Integration and Query Processing

The system uses Ollama to deploy LLMs locally so the user can have complete privacy with data and perform more efficient analysis operations at lower latencies. The implementation supports several open-source models, for example, some of the tested models are Llama, Falcon, and Mistral with 8 to 32 billion parameters which are optimized for dual NVIDIA TITAN RTX GPU infrastructure. This architecture also includes a smart query processing pipeline in which contextual information retrieved from the vector store is combined with the reasoning capabilities of LLMs running locally. It offers a high-throughput pipeline that is seamlessly integrated and compliant with strict data privacy policies. Using Ollama for local processing, the system resolves some of the most important security concerns while providing flexibility in model selection and configuration.

4.3 Development Environment and Infrastructure

The proposed security analysis system should be implemented in a well-organized development environment that maintains a balance between computational requirements, security considerations, and system efficiency. This section discusses the detailed architecture that supports the system's core capabilities.

4.3.1 Hardware Infrastructure and Resource Management

The computational architecture of this suggested system rests on high-performance GPU computing, which is needed to run efficient local LLM operations and embedding creation. The core hardware configuration consists of:

- NVIDIA TITAN RTX Dual GPU with 24GB VRAM Each
- High-performance CPU
- RAM Requirements:
  - 7B models: At least 8 GB of RAM.
  - 13B models: At least 16 GB of RAM.
  - 33B models: At least 32 GB of RAM.
  - 70B models: At least 64 GB of RAM.

4.3.2 Software Stack and Dependencies

To ensure scalability, performance, and ease of maintenance, the proposed system will be implemented using open-source tools, libraries, and frameworks. The main software components are:

Listing 4.3: Core Dependencies

```
Core Dependencies:


- langchain==0.1.10  # Framework for LLM operations
- chromadb           # Vector storage
- streamlit          # User interface framework
- ollama             # Platform for running and managing LLMs
   locally
- python-dotenv      # Environment configuration
- pyshark            # PCAP file processing
- tqdm               # Progress monitoring
- langchain-community  # LangChain capabilities
```

Additional specialized libraries for specific functionalities are:

Listing 4.4: Additional Processing Libraries

```
Processing Libraries:


- JSONLoader          # Data parsing
- RecursiveCharacterTextSplitter  # Content segmentation
- OllamaEmbeddings  # Local embedding generation, it generates
    embeddings using Ollama's models.
```

Here are the details of the technology stack and implementation:

- **Data Preprocessing:**

  - `PyShark`: For PCAP parsing and feature extraction
  - `Python-dotenv`: For managing environment variables and API keys
  - `Requests`: For "VirusTotal" API integration
  - `Tqdm`: For progress tracking during data preprocessing

- **Vector Database:**
  - `ChromaDB`: For vector embedding storage and similarity search
  - `OLLAMA Embeddings`: For vector embedding generation
  - `LangChain`: For integrating OLLAMA with ChromaDB
  - Handling large JSON file:
    - `JSONLoader`: Structured data parsing
    - `RecursiveCharacterTextSplitter`: Content segmentation

- **LLM Query Processing and Response Generation:**
  - `OLLaMA`: For query understanding and response generation using LLMs locally
  - `LangChain`: Framework that helps to integrate LLMs into applications.
  - `Groq`: Alternative of Local implementation using Groq API (optional)

- **User Interface:**
  - `Streamlit`: For building the web-based user interface
  - `Streamlit Components`: For integrating custom UI elements and visualizations

5 Testing and Evaluation

5.1 Performance Analysis

A detailed evaluation and testing were done on the proposed system with captured network traffic files (PCAP) and security incident scenarios for its effectiveness and performance assessment.

5.1.1 Experimental Setup and Datasets

The datasets used for testing and evaluation are collected from the website "Malware Traffic Analysis" [67], an open-source repository of real-world malware network traffic data. This repository contains a large dataset of PCAP files with different attack signatures. In the evaluation, these PCAP files were used to test the proposed system. The system was deployed in a high-performance computational environment to make it robust and reliable. This configuration was specifically selected to accommodate the high performance and memory needed to run those LLMs. The experimental setup is equipped with the following specifications detailed in Table 5.1:

The datasets were carefully chosen to cover a wide range of security incidents, including malware infections, for example, ransomware, trojans, botnets, phishing, and social engineering attacks. A total of Twelve PCAP files were chosen from the "Malware Traffic Analysis" website to provide a diverse range of attack types

Table 5.1: System Specifications

Component	Specification
Processing Unit	AMD Ryzen Threadripper 2920X 12-Core Processor
Memory	96GB DDR4
Storage	1TB NVMe SSD
GPU	2 NVIDIA TITAN RTX GPUs with 24GB memory each
Operating System	Ubuntu 22.04.3 LTS

and network traffic characteristics. The testing dataset composition was distributed across different malware categories in order to allow for comprehensive system testing. Each of the files is processed individually using the developed pipeline, which includes PCAP parsing, feature extraction, and threat intelligence enrichment. The preprocessed data is then stored in the ChromaDB vector database for efficient similarity search and retrieval. Table 5.2 presents the distribution of malware families in our test dataset.

Table 5.2: Malware Category Distribution

Malware Category	Number of Samples	Percentage	Actual File Size
Ransomware	5	41.67%	20MB
Banking Trojans	4	33.33%	12MB
Other Malware	3	25.00%	15MB

The system comes with a very user-friendly interface where users can upload PCAP files and analyze them with the help of LLM. They can chat with the model, and it will give insightful information and recommendations based on the captured traffic file. The bellow Figure 5.1 is the User Interface of the system:

5.1.2 Prompt Design

The system prompt plays an important role in steering this LLM-powered system towards giving relevant and correct answers to user queries. Listing 5.1 is the example prompt that is used in the evaluation process, it demonstrates how a well-defined prompt provides the necessary context for the LLMs to reasonably analyze packet captures. The prompt begins by stating that the assistant is an expert in packet

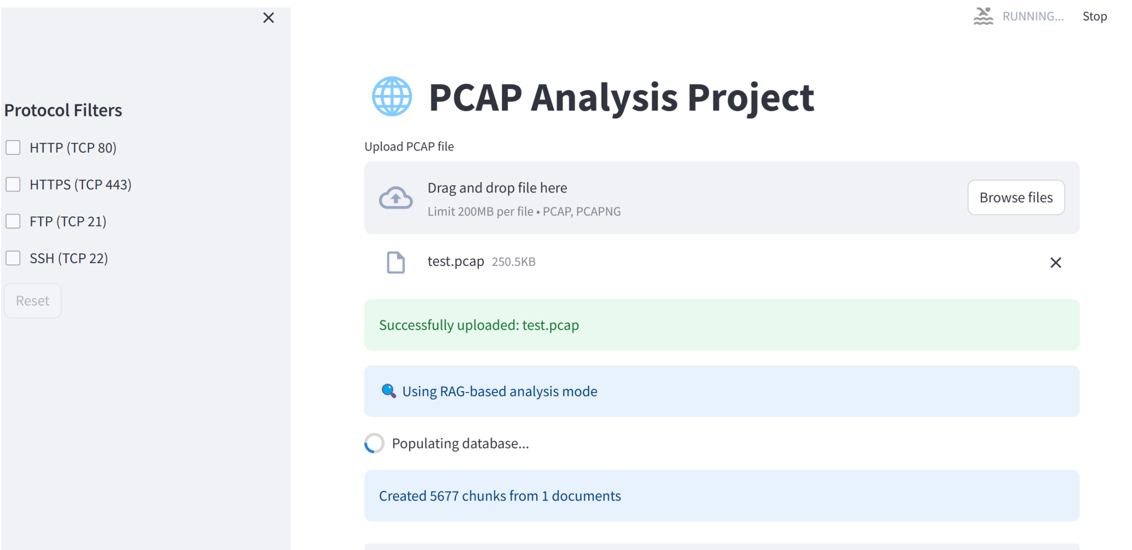

Figure 5.1: User Interface

capture analysis. This improves the LLMs' ability to analyze network traffic and respond in ways that are relevant to the needs of the Network Engineer and Cybersecurity professional.

It also has detailed instructions for varied user queries. For instance, if the query is about an application layer protocol such as HTTP, HTTPS, or SSH, the prompt tells the LLMs to search the database for the corresponding port numbers. This allows the system to quickly locate accurate information and deliver protocol-specific relevance. For general analysis, the prompt encourages the LLMs to pull information from all layers of the network stack (Ethernet, IP, transport, etc) with source and destination IPs, port numbers, and so on, highlighted as critical information to return.

At runtime, the `{context}` placeholder for the given prompt is dynamically filled with real packet capture information retrieved from the vector and VirusTotal databases, allowing the LLMs to access relevant network traffic for context-based responses. Similarly, the user's question replaces the placeholder `{question}`. When the prompt is about a topic unrelated to the context provided, it instructs the LLMs to inform the user that the prompt is unrelated to the context while still providing relevant information from its general knowledge. That keeps the system useful and responsive, even for irrelevant questions.

Listing 5.1: Prompt Design

```
You are a helper assistant specialized in analyzing packet
   captures used for technical analysis and malware detection.
    Use the information present in the vector database and
   VirusTotal analysis results to answer all questions
   truthfully.

When analyzing security-related questions or potential threats
   :
1. First check if there are any VirusTotal results in the
   context
2. Consider both the packet capture data and VirusTotal
   analysis when forming your response
3. If malicious activity is detected, explain the nature of
   the threat and potential implications
4. Provide specific indicators or patterns found in the packet
    capture that correlate with the VirusTotal findings

For general analysis:
- Extract information about every layer (such as ethernet, IP,
    transport layer)
- Identify source and destination IPs, port numbers and other
   possible insights
- Note any suspicious patterns or known malicious indicators
- Correlate findings with any VirusTotal security analysis
```

```
   results

Format your response in markdown text with line breaks. You
   are encouraged to use emojis to make your response more
   presentable and fun.

hints:
http means tcp.port = 80
https means tcp.port = 443
ssh means tcp.port = 22

{context}

---

Answer the question based on the above context and use your
   own knowledge to provide a comprehensive explanation: {
   question}

Note: if the asked question is not relevant to the context,
   inform the user that it is not relevant to the provided
   documents and then use your own knowledge to answer it.
```

5.1.3 Evaluation Metrics and Results

The following key metrics are used to evaluate the performance of the LLM-Powered network traffic analysis system:

- Accuracy: This measures how well the system correctly identifies each type of PCAP file from the test dataset and classifies them into the right categories.

It is calculated as the percentage of correctly identified events out of the total dataset.

- Precision and Recall: Precision shows how many of the system's positive predictions are correct, while recall measures how many actual positive events were correctly identified. Together, these metrics give a clearer picture of the system's performance by accounting for both false positives and false negatives.

- Query Response Time: This tracks how quickly the system processes user queries, retrieves information from the vector database, and generates AI-powered responses. It is one of the key metrics for evaluating the system's ability to support fast incident responses.

- Model Comparison: Performance of open-source LLMs: LLAMA, Falcon, and Mistral with various parameters (7B, 12B, etc.). The comparison reveals each model's effectiveness for this specific task.

5.1.3.1 Vector Database Performance Analysis

Vector databases' performance is measured using several key metrics, including the time it takes to create an embedding, the efficiency of initial indexing, the query response time, and the similarity search quality. For this testing, two open-source models were used: "nomic-embed-text" and "mxbai-embed-large". The "nomic-embed-text model" has 137 million parameters and is one of the smaller models evaluated. However, according to the model's performance benchmark, it outperforms OpenAI's "Ada-002" and "text-embedding-3-small" models in both short and long context tasks. On the other hand, "mxbai-embed-large" is a slightly larger model with 334 million parameters.

To obtain the results, both models were used for embedding creation and a query response task. Each model was used to embed all of those PCAP files separately, and performance was evaluated in four categories: embedding creation time, initial indexing time, query processing time, and query response quality. Both models had consistent GPU memory spikes when it was creating the embedding. The "nomic-embed-text" model was using 1818 MB of GPU memory during the entire runtime to process it whereas the "mxbai-embed-large" model was using 1224 MB. Despite having more than twice as many parameters, the "mxbai-embed-large" model surprisingly used less GPU RAM. Figure 5.2 5.3 shows the GPU memory consumption of both models during the task running process. *"watch -n 1 nvidia-smi"* command was used to view track memory uses during the processing time.

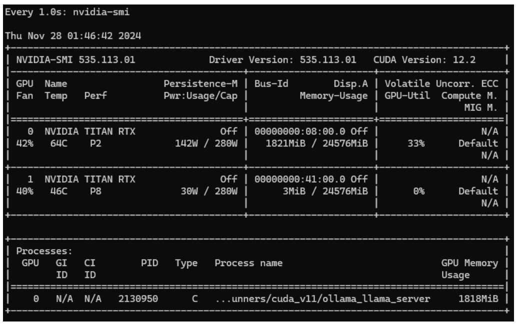

Figure 5.2: Memory Consumption of "nomic-embed-text" Model

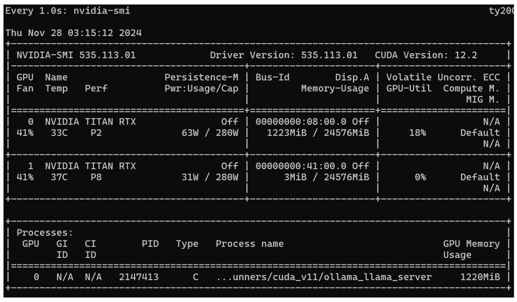

Figure 5.3: Memory Consumption of "mxbai-embed-large" Model

In terms of preliminary indexing and query processing tasks, the models both performed well and processed efficiently and quickly. However, a significant difference appeared in the time spent creating the embeddings. However, when we analyzed embedding creation time, the difference was significant. The "mxbai-embed-large" model outperformed the "nomic-embed-text" model and efficiently converted those document inputs into the vector.

Table 5.3 provides a breakdown of the above comparison, including GPU memory consumption, embedding creation time, indexing time, and query return time. Overall "mxbai-embed-large" model performed better in the embedding task.

Table 5.3: Vector Database Performance Comparison: "nomic-embed-text" and "mxbai-embed-large"

Operation Phase	Model	Average Time (ms)	Memory Usage (MB)
Embedding Creation	nomic-embed-text	205000	1818
	mxbai-embed-large	167000	1224
Initial Indexing	nomic-embed-text	340	1818
	mxbai-embed-large	300	1224
Query Processing	nomic-embed-text	300	1818
	mxbai-embed-large	280	1224

5.1.3.2 Model Performance Evaluation

To evaluate the performance of different LLMs, a set of 12 PCAP files was used to carry out a number of tests. Each PCAP file was tested using the selected LLMs, such as Llama2-13B, Llama3.1-8B, Mistral-7B, and Falcon-40B. During the evaluation phase, the PCAP files were uploaded one by one, and these models' performance was tested. The performance of each LLM was evaluated according to its ability to correctly identify relevant patterns and provide relevant information based on user queries. In Table 5.4, the model performance analysis summary is presented, including the accuracy (the proportion of correct responses), response time, and memory usage for each LLM.

Table 5.4: Comprehensive Model Performance Analysis

LLM	Accuracy (Proportion)	Response Time (s)	Memory Usage (GB)
Llama2-13B	0.75	0.9	14.5
Llama3.1-8B	0.67	0.8	7.1
Mistral-7B	0.67	0.8	7.2
Falcon-40B	n/a	2.5	31.1

The results show that Llama2-13B produced the highest accuracy out of all models, with relevant and actionable information being identified in 75% of the tested PCAP files (9 out of 12). Llama3.1-8B and Mistral-7B achieved slightly lower accuracies of 67%.

During testing, each model demonstrated various response styles, which could be due to the prompt template. All models used the same prompt template. However, differing prompt templates cause the LLMs to respond differently. Mistral-7B delivered concise, focused solutions that immediately answered the user's inquiries. In comparison, Llama2-13B produced more thorough responses, including context and insights. Falcon-40B's performance was not formally examined due to its excessive memory consumption. It is evident that larger models, such as the Falcon-40B, have more capabilities, but their implementation in resource-constrained contexts can be difficult.

Among those tested LLMs in this study, Llama2-13B achieved the best performance based on the evaluation results. Compared to other models, better accuracy and response generation make it a good candidate for a tactical analyst to sort through network traffic or perform incident response. However, model size plays a very critical role in performance, even though Llama3.1 is the latest model of Llama series it was outperformed by LLama2 which is mostly likely because of the size of the model. Llama3.1 was tested with an 8B parameter version, whereas Llama2-13B used a larger 13B parameter version. Here are two screenshots of the system UI, shown in Figure 5.4 and Figure 5.5, where the user was chatting with the PCAP file and retrieving insightful information directly from it. The user's questions included queries such as, "Which IPs are making HTTP requests?" and "For the HTTP requests identified, what are the source and destination IPs, and what data was exchanged?"

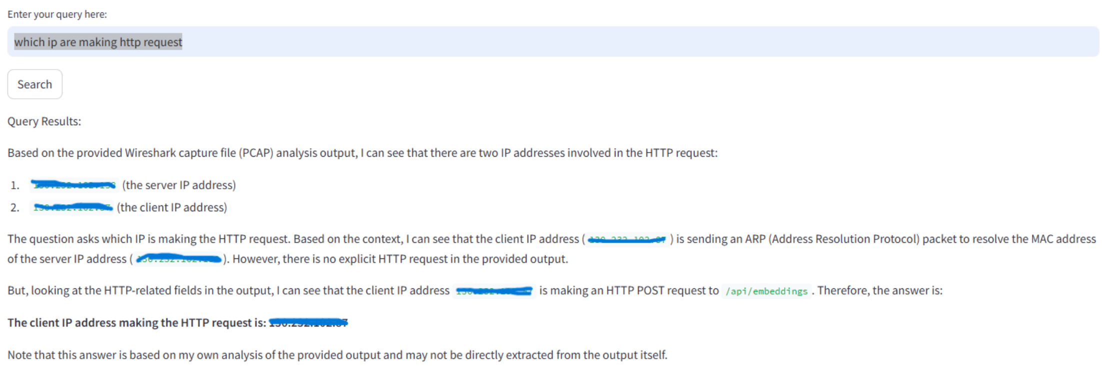

Figure 5.4: Interactive System UI Showing User Queries

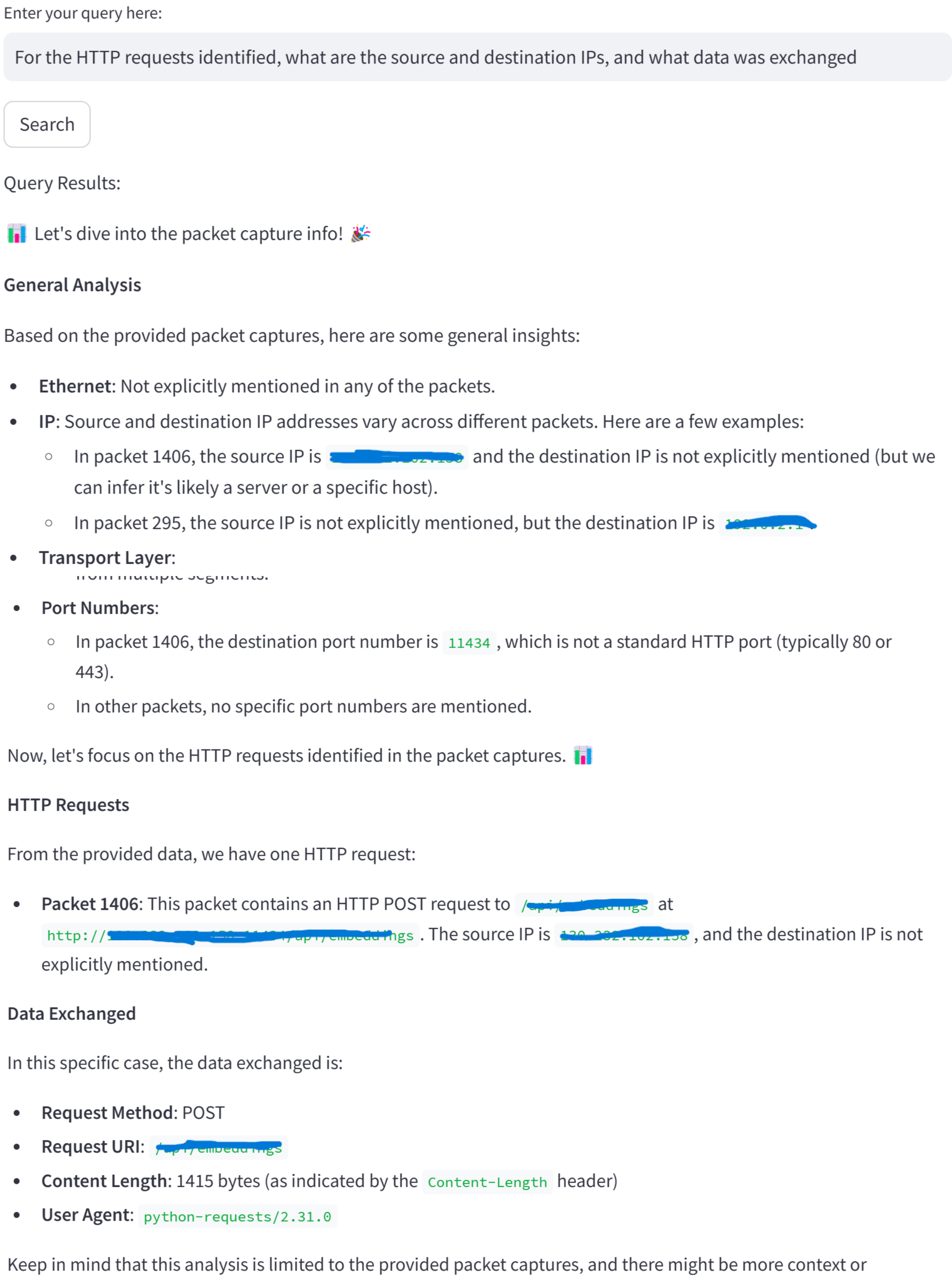

Figure 5.5: Interactive System UI Showing User Queries

6 Conclusion and Future Work

6.1 Summary of Findings and Contributions

This thesis proposes a new framework for cybersecurity analysis: a privacy-preserving, LLM-powered system that is capable of analyzing network traffic. By integrating locally run large language models with open-source security tools, this privacy-centric design maintains high-fidelity analysis without compromising data confidentiality. This solves important aspects of RQ1, which has been examined extensively in Chapter 2 with respect to traditional approaches.

This work contributes in several ways. To begin, a complete pipeline comprising PCAP parsing, feature extraction, and threat knowledge enrichment was implemented with state-of-the-art tools and libraries such as PyShark and VirusTotal. This whole system is designed to run locally, ensuring that user has full control of their data and that sensitive information regarding network traffic (privacy, security, anonymity) is preserved. The machine operates on the principle of "data stays where it belongs" as it uses open-source LLMs and runs proprietary hardware, which limits the potential for data leakage, and also ensures confidentiality throughout the investigative process. This alone is a meaningful contribution because it addresses increasing concerns surrounding the privacy and safety of AI-based digital forensic investigations. The implementation details introduced in Chapter 4 target the most crucial components for both RQ1 and RQ2.

The second contribution demonstrates how integrating a vector database can improve the storage and retrieval of security-related embeddings, allowing for context-aware threat detection. For example, using similarity search in a custom-created vector database improves both response accuracy and retrieval speed. This complements the second part of RQ2, which focuses on cost-effective and customizable solutions.

The third contribution is to show the benefits of using an external threat intelligence database from the VirusTotal API, which gives LLM recently updated information and helps to produce more accurate intuition. The effectiveness mentioned in RQ3 is supported by the experimental results presented in Chapter 5.

6.2 Reflection on the Practicality and Effectiveness

The LLM-enhanced digital forensics system was developed and tested. This work sheds light on how vector databases and LLMs, as a platform, can be used in practical cybersecurity applications. This approach allows for automated feature extraction and data parsing, as well as other manual or time-consuming tasks associated with network traffic analysis. It also has room to improve in terms of the correlation of events across multiple data sources. This allows security analysts to concentrate on higher-level decision-making, and incident response, thus increasing the efficiency of digital forensics investigations by leveraging smart algorithms and large language models.

However, it is critical to understand its limitations and challenges. A particularly problematic issue is the interpretability and the explainability of the produced insights and recommendations. Although LLM has demonstrated remarkable performance in generating coherent and contextually relevant responses, there is still some ambiguity about the reasoning mechanisms and decision-making criteria inherent in these models. It is important to have some sort of assurance that the output of a system is accurate, which can be a problem, especially for big cybersecurity investigations because false positives or negatives can be very costly.

Also to run it locally, it requires a lot of computer power. Only small models were tested due to hardware limitations and that may limit system performance during the evaluation phase. Larger models, with more parameters and more training data, are expected to produce better results and more accurate and nuanced information. Therefore, it should be noted that LLM models can be computationally intensive to run, so organizations should pay close attention to the old performance versus cost dynamic with an LLM-infused traffic analysis system.

6.3 Further Research and Improvements

Even though the system has produced promising results, there are still areas where it can be improved for more practical use. One important area for future research would be to fine-tune open-source LLMs to handle specific security tasks. While current models perform reasonably well, fine-tuning the model with a customized dataset for cybersecurity purposes could significantly improve its accuracy. Fine-tuning allows the model to learn the artifacts and patterns specific to this domain that will provide recommendations more precisely around incident response and forensic analysis. As described in the recent cybersecurity literature review paper by Xu *et al.* [7], domain-specific fine-tuning can significantly improve model performance in specialized tasks.

One other challenge of testing was that there was also a limited capacity from the hardware side. This situation really constrains the usage of bigger LLMs because of this limitation only smaller models were tested. However, with better availability of hardware, larger models such as LLaMA-70B and other similar models could give even better results. So future research will be done with higher computational resources so that the performance of those bigger models can also be tested. Notably, larger models with more training data and parameters are expected to produce better outcomes even without fine-tuning. Another topic for future work will be the addition of more external sources of data and threat intelligence feeds. For example, endpoint logs, system events, user behavior, or external threat intelligence feeds may make possible much deeper cybersecurity analytics. This will help in incident reconstruction and detailed attacker behavior understanding.

References

[1] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, H. Anderson, H. Roff, G. C. Allen, J. Steinhardt, C. Flynn, S. Ó. hÉigeartaigh, S. Beard, H. Belfield, S. Farquhar, C. Lyle, R. Crootof, O. Evans, M. Page, J. Bryson, R. Yampolskiy, and D. Amodei, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation", *arXiv preprint arXiv:1802.07228*, 2018.

[2] N. Kaloudi and J. Li, "The AI-based cyber threat landscape: A survey", *ACM Computing Survey*, vol. 53, no. 1, pp. 1–34, Feb. 2020.

[3] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey", *Computer Communications*, vol. 170, pp. 19–41, 2021.

[4] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via BERT", in *2021 International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, 2021, pp. 1–8.

[5] W. Fu, K. Yang, R. G. Dutta, X. Guo, and G. Qu, "LLM4SecHW: Leveraging domain-specific large language model for hardware debugging", in *2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Tianjin, China: IEEE, Dec. 2023.

[6] I. Elsharef, Z. Zeng, and Z. Gu, “Facilitating threat modeling by leveraging large language models”, in *Workshop on AI Systems with Confidential Computing (AISCC) 2024*, San Diego, USA, 2024.

[7] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang, “Large language models for cyber security: A systematic literature review”, *arXiv preprint arXiv:2405.04760*, 2024.

[8] D. Myers, R. Mohawesh, V. I. Chellaboina, A. L. Sathvik, P. Venkatesh, Y.-H. Ho, H. Henshaw, M. Alhawawreh, D. Berdik, and Y. Jararweh, “Foundation and large language models: Fundamentals, challenges, opportunities, and social impacts”, *Cluster Computing*, vol. 27, no. 1, pp. 1–26, 2024.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need”, in *Advances in Neural Information Processing Systems*, vol. 30, Long Beach, CA, USA, 2017.

[10] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, T. Lestable, and N. S. Thandi, “Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for IoT/IIoT devices”, *IEEE Access*, vol. 12, pp. 23 733–23 750, 2024.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding”, *arXiv preprint arXiv:1810.04805*, 2018.

[12] OpenAI, *OpenAI platform: Models documentation*, Accessed: 2024-11-02, 2024. [Online]. Available: `https://platform.openai.com/docs/models`.

[13] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner,

S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners", *arXiv preprint arXiv:2005.14165*, 2020.

[14] M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, and N. Tihanyi, "Generative AI and large language models for cyber security: All insights you need", *arXiv preprint arXiv:2405.12750*, 2024.

[15] B. Ghosh, "Empowering language models: Pre-training, fine-tuning, and in-context learning", *Medium*, Jun. 2023, Accessed: 2024-11-26. [Online]. Available: `https://medium.com/@bijit211987/the-evolution-of-language-models-pre-training-fine-tuning-and-in-context-learning-b63d4c161e49`.

[16] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly", *High-Confidence Computing*, vol. 4, no. 2, pp. 100–211, 2024.

[17] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey", *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020.

[18] F. N. Motlagh, M. Hajizadeh, M. Majd, P. Najafi, F. Cheng, and C. Meinel, "Large language models in cybersecurity: State-of-the-art", *arXiv preprint*, 2024.

[19] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, "Nist special publication 800-61, revision 2: Computer security incident handling guide", *NIST, US Department of Commerce*, 2012.

[20] A. Ahmad, J. Hadgkiss, and A. Ruighaver, "Incident response teams – challenges in supporting the organisational security function", *Computers & Security*, vol. 31, no. 5, pp. 643–652, 2012.

[21] S. Rao, "After the breach: Incident response within enterprises", *arXiv preprint arXiv:2406.07559*, 2024.

[22] F. R. Alzaabi and A. Mehmood, "A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods", *IEEE Access*, vol. 12, pp. 30 907–30 927, 2024.

[23] Verizon, "2023 data breach investigations report", Verizon, Tech. Rep., 2023, Accessed: 2024-11-03. [Online]. Available: `https://www.verizon.com/business/resources/T76/reports/2023-data-breach-investigations-report-dbir.pdf`.

[24] M. Bartnes, N. B. Moe, and P. E. Heegaard, "The future of information security incident management training: A case study of electrical power companies", *Computers & Security*, vol. 61, pp. 32–45, 2016.

[25] K. Mandia, M. Pepe, and J. Luttgens, *Incident Response & Computer Forensics, Third Edition*. McGraw-Hill Education, 2014, ISBN: 9780071798686.

[26] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix", *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[27] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security", in *10th international conference on cyber Conflict (CyCon)*, IEEE, Tallinn, Estonia, 2018, pp. 371–390.

[28] S. C. Sundaramurthy, J. McHugh, X. Ou, M. Wesch, A. G. Bardas, and S. R. Rajagopalan, "Turning contradictions into innovations or: How we learned to stop whining and improve security operations", in *Proceedings of the Twelfth*

USENIX Conference on Usable Privacy and Security, Denver, CO, USA, 2016, pp. 237–251.

[29] S. Freitas and A. Gharib, "Graphweaver: Billion-scale cybersecurity incident correlation", in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, Boise, ID, USA, 2024, pp. 4479–4486.

[30] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures", *Sensors*, vol. 21, no. 14, 2021.

[31] Y. Wang, H. Liu, Z. Li, Z. Su, and J. Li, "Combating advanced persistent threats: Challenges and solutions", *IEEE Network*, vol. 38, pp. 324–333, 2024.

[32] E. Casey, *Digital evidence and computer crime: Forensic science, computers, and the internet.* Academic press, 2011.

[33] S. Thakare, A. Pund, and M. A. Pund, "Network traffic analysis, importance, techniques: A review", in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2018, pp. 376–381.

[34] N. Alqudah and Q. Yaseen, "Machine learning for traffic analysis: A review", *Procedia Computer Science*, vol. 170, pp. 911–916, 2020.

[35] A. Dhakad, S. Singh, Mohana, M. Moharir, and A. K. A. R, "Real time network traffic analysis using artificial intelligence, machine learning and deep learning: A review of methods, tools and applications", in *2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, Erode, India, 2023, pp. 372–378.

[36] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, "Automatic root cause analysis via large language models

for cloud incidents", in *Proceedings of the Nineteenth European Conference on Computer Systems*, Athens, Greece, 2024, pp. 674–688, ISBN: 9798400704376.

[37] Y. Chen, Z. Ding, L. Alowain, X. Chen, and D. Wagner, "Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection", in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, Hong Kong, China, pp. 654–668.

[38] M. Fu and C. Tantithamthavorn, "Linevul: A transformer-based line-level vulnerability prediction", in *Proceedings of the 19th International Conference on Mining Software Repositories*, Pittsburgh, Pennsylvania, 2022, pp. 608–620.

[39] F. Demirkıran, A. Çayır, U. Ünal, and H. Dağ, "An ensemble of pre-trained transformer models for imbalanced multiclass malware classification", *Computers & Security*, vol. 121, p. 102 846, 2022.

[40] R. J. Joyce, T. Patel, C. Nicholas, and E. Raff, "Avscan2vec: Feature learning on antivirus scan data for production-scale malware corpora", in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 185–196.

[41] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system", *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022.

[42] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "Chatspamdetector: Leveraging large language models for effective phishing email detection", *arXiv preprint arXiv:2402.18093*, 2024.

[43] S. Jamal, H. Wimmer, and I. H. Sarker, "An improved transformer-based model for detecting phishing, spam and ham emails: A large language model approach", *SECURITY AND PRIVACY*, vol. 7, no. 5, e402, 2024.

[44] P. Evangelatos, C. Iliou, T. Mavropoulos, K. Apostolou, T. Tsikrika, S. Vrochidis, and I. Kompatsiaris, "Named entity recognition in cyber threat intelligence using transformer-based models", in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, Rhodes, Greece, 2021, pp. 348–353.

[45] B. Ahmad, S. Thakur, B. Tan, R. Karri, and H. Pearce, "On hardware security bug code fixes by prompting large language models", *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 4043–4057, 2024.

[46] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu, "Harnessing the power of LLMs in practice: A survey on chatgpt and beyond", *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 6, Apr. 2024.

[47] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, *et al.*, "Gpt-neox-20b: An open-source autoregressive language model", *arXiv preprint arXiv:2204.06745*, 2022.

[48] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, *Carbon emissions and large neural network training*, 2021.

[49] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, New York, USA, 2020, pp. 13 693–13 696.

[50] J. Zhang, H. Bu, H. Wen, Y. Chen, L. Li, and H. Zhu, "When LLMs meet cybersecurity: A systematic literature review", *arXiv preprint arXiv:2405.03644*, 2024.

[51] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (LLMs): A survey", *arXiv preprint arXiv:2403.05156*, 2024.

[52] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, *et al.*, “Extracting training data from large language models”, in *30th USENIX Security Symposium (USENIX Security 21)*, Vancouver, B.C., Canada, 2021, pp. 2633–2650.

[53] Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, *et al.*, “Prompt injection attack against LLM-integrated applications”, *arXiv preprint arXiv:2306.05499*, 2023.

[54] Y. Liu, Y. Yao, J.-F. Ton, X. Zhang, R. G. H. Cheng, Y. Klochkov, M. F. Taufiq, and H. Li, “Trustworthy LLMs: A survey and guideline for evaluating large language models’ alignment”, *arXiv preprint arXiv:2308.05374*, 2023.

[55] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, “Explainability for large language models: A survey”, *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, Feb. 2024.

[56] S. Abdali, R. Anarfi, C. Barberan, and J. He, “Securing large language models: Threats, vulnerabilities and responsible practices”, *arXiv preprint arXiv:2403.12503*, 2024.

[57] B. Ahmad, B. Tan, R. Karri, and H. Pearce, “FLAG: Finding line anomalies (in code) with generative AI”, in *In proceedings Network and Distributed System Security (NDSS) Symposium*, San Diego, CA, USA.

[58] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, and Y. Liu, “Masterkey: Automated jailbreaking of large language model chatbots”, in *Proceedings 2024 Network and Distributed System Security Symposium*, ser. NDSS 2024, Internet Society, 2024.

[59] M. Gurman, “Samsung bans employees from using AI after spotting ChatGPT data leak”, *Business Standard*, May 2023, Accessed: 2024-11-07. [Online]. Available: `https://www.business-standard.com/technology/tech-news/sams`

ung-bans-employees-from-using-ai-after-spotting-chatgpt-data-leak-123050200197_1.html.

[60] T. Gao and P. Ji, "Extended abstract: Leveraging large language models to identify Internet censorship through network data", in *Free and Open Communications on the Internet*, 2024.

[61] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs", *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[62] Chroma, *Getting started*, Accessed: 2024-11-24, 2024. [Online]. Available: https://docs.trychroma.com/getting-started.

[63] LangChain, *Langchain: Build context-aware reasoning applications*, Accessed: 2024-11-24, 2024. [Online]. Available: https://github.com/langchain-ai/langchain.

[64] KimiNewt, *Pyshark: Python packet parser using wireshark's tshark*, Accessed: 2024-11-24, 2024. [Online]. Available: https://kiminewt.github.io/pyshark/.

[65] VirusTotal, *Virustotal api v3 overview*, Accessed: 2024-11-24, 2024. [Online]. Available: https://docs.virustotal.com/reference/overview.

[66] D. Crockford and C. Morningstar, *Standard ecma-404 the json data interchange syntax*, Dec. 2017.

[67] B. Duncan, *Malware-traffic-analysis.net*, Accessed: 2024-11-27, 2024. [Online]. Available: https://malware-traffic-analysis.net/.