A 5G Security Recommendation System Based on Multi-Modal Learning and Large Language Models

Jia Wei Yao

A Thesis

in

The Department

of

Concordia Institute for Information System Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Information System Security) at

Concordia University

Montréal, Québec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Jia Wei Yao**

Entitled: **A 5G Security Recommendation System Based on Multi-Modal Learning and Large Language Models**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information System Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Jeremy Clark

______________________________ Examiner
Dr. Arash Mohammadi

______________________________ Supervisor
Dr. Lingyu Wang

______________________________ Co-supervisor
Dr. Suryadipta Majumdar

Approved by ______________________________
Chun Wang, Chair
Department of Concordia Institute for Information System Engineering (CIISE)

__________ 2023

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

A 5G Security Recommendation System Based on Multi-Modal Learning and Large Language Models

Jia Wei Yao

Deploying 5G networks on top of cloud-native environments provides unique benefits including cost-effectiveness, flexibility, and scalability. However, the increased complexity of a cloud-native 5G deployment also brings new security challenges to existing solutions for security monitoring and security auditing. A security analyst may need to analyze events coming from various sources, such as security monitoring (e.g., Falco) and security auditing (e.g., Kubescape) systems deployed at both the 5G and cloud (container) levels. Understanding the relationships between events coming from those different sources is usually challenging since those security solutions may have very different monitoring/auditing criteria and scopes. Relying on manual analysis and domain knowledge may also be too slow and error-prone considering the sheer scale of a cloud-native 5G deployment. In this paper, we propose 5GSecRec, a 5G security recommendation system that leverages multi-modal learning to correlate alerts from four aspects of a cloud-native 5G deployment, i.e., security monitoring and security auditing systems, deployed at both 5G/Kubernetes® levels. Also, 5GSecRec further eases security analysts' job by answering their questions expressed in a natural language (e.g., "What is the impact of a Kubernetes alert on the 5G level?") using large language models (LLMs) fine-tuned with the learned knowledge about correlated alerts. We implement 5GSecRec based on free5GC, Kubernetes, and LLMs from HuggingFace, and our experimental results demonstrate the effectiveness of our solution (e.g., up to 89.5% of correlation accuracy, and comparable question-answering performance to ChatGPT but without data confidentiality and privacy concerns).

Acknowledgments

I would like to express my sincere gratitude to Dr. Suryadipta Majumdar and Dr. Lingyu Wang for their expert guidance in my research on 5G network security. Their mentorship has been extremely essential in the early stages of my research career, providing me with the foundation and confidence to tackle research challenges. I am also thankful to my labmate and colleague, Md Nazmul. My appreciation extends to all members of the Concordia Information Security Research (CISR) Lab for fostering a collaborative and dynamic team environment. Finally, I am deeply grateful to my parents for their unwavering support and respect for my career choices.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This chapter describes the context and problem, discusses the research gap, presents our motivation, outlines our solution, and provides an overview of the thesis organization.

1.1 Context and Problem Statement

The cloud-native technology (i.e., the design, implementation, and deployment of applications in cloud environments [1]) can empower telecommunication providers to provide more cost-effective, flexible, and scalable 5G services (e.g., AWS is used to deploy an entire cloud-native 5G network [2] and VMware Telco Cloud platform is designed for a similar purpose [3]). A cloud-native 5G service deployment is typically more complex as it may involve many complementary technologies, such as Network Function Virtualization (NFV) [4], containerization (e.g., Docker [5]), orchestration (e.g., Kubernetes® [6]), etc. Those technologies work together to support a cloud-native 5G service deployment at two major abstraction levels, i.e., the *service level* that contains virtual 5G network functions [7] (e.g., Unified Data Management (UDM), Access Mobility Management Function (AMF), and User Plane Function (UPF)), and the *virtualization level* which abstracts the underlying virtual infrastructure in terms of containers to provide isolated compute network and storage resources.

The increased complexity of a cloud-native 5G deployment brings novel security challenges. Security analysts would now face an overwhelming amount of events coming from both the service level and virtualization level, such as alerts raised by security monitoring (e.g., Falco [8]) and security auditing (e.g., Kubescape [9]) tools. Understanding the relationships between events coming from different levels (service and virtualization) and from different tools (monitoring and auditing) can usually provide critical insights about an attack, e.g., how a 5G function may be impacted by an underlying container-related attack, and whether a policy breach (reported by Kubescape) has been exploited in attacks (detected by Falco). However, correlating events between those different levels and tools can be challenging due to the inherent complexity of a cloud-native 5G deployment and the fact that different security solutions may have incompatible criteria and scopes. The sheer scale of a cloud-native 5G deployment also means that relying on security analysts to perform manual analysis based on their domain knowledge would be too slow and error-prone.

1.2 Research Gap

Existing solutions on event correlation (e.g., [10–12]) typically rely on predefined rules or dependency graphs, which necessitates significant human intervention and domain knowledge. Furthermore, the scale and complexity of a cloud-native 5G deployment may render rule-based approaches practically infeasible, and the self-service and highly dynamic nature of the virtual environment could also make it difficult to keep up with the need for constantly updating rules. Additionally, those existing solutions usually stop at correlated events, leaving a gap between the correlations and their interpretation for answering actual questions from security analysts. To that end, most recent efforts on question-answering systems [13–15] fall short since they prioritize knowledge retrieval over cause-consequence analyses (which are often required by security analysts). Finally, although there exist powerful question-answering systems like ChatGPT [16], a 5G operator may be reluctant to adopt them directly due to well-known concerns about data privacy and confidentiality [17, 18].

1.3 Motivation

Figure 1 illustrates an attack on a cloud-native 5G deployment (bottom) and the challenges faced by a security analyst toward understanding the attack (top). Specifically, the deployment is composed of a service level (containerized 5G core network) and a virtualization level (Kubernetes container cluster). The attack exploits a Kubernetes vulnerability (CVE-2023-3893 [19]) to gain privilege escalation to launch DoS attacks on the 5G network functions. In analyzing the attack, the security analyst is overwhelmed by a large number of alerts and breaches separately reported by the security monitoring and auditing tools deployed at both the 5G and Kubernetes levels. First, it requires significant manual effort and domain knowledge for the analyst to understand the correlations between those alerts and breaches, both across different levels and across different tools. Second, even if such correlations can be obtained, it would still be challenging to interpret them for answering practical questions, e.g., whether the highlighted K8S-level alert *Created Privileged Containers* indicates the exploitation of any existing K8S-level breaches, whether the alert relates to, or impact 5G level, and whether this alert has any supporting evidence reported by other tools. Finally, as shown on the right, although Large Language Models (LLMs) like chatGPT [16] are promising for understanding and answering users' questions, data confidentiality and privacy concerns may prevent the 5G operator from sending data to third-party LLMs.

1.4 Our Solution

To address those limitations, we propose *5GSecRec*, a 5G security recommendation system based on multi-modal learning and LLM. Specifically, our ideas are twofold. First, by regarding the four aspects of a cloud-native 5G deployment (i.e., security monitoring and security auditing systems, deployed at both 5G/Kubernetes levels) as different modalities, 5GSecRec leverages multi-modal learning to correlate events (alerts and breaches) from different aspects of the deployment. For instance, an unexpected connection alert from the Kubernetes Monitoring System and an abnormal traffic alert from the 5G Monitoring System might be correlated, indicating a coordinated attack. Second, 5GSecRec further eases security analysts' job by applying such learned knowledge about correlations to fine-tune an LLM, such that analysts may directly query 5GSecRec in a natural language (e.g., "What is the impact of a Kubernetes alert on the 5G level?"). This LLM is hosted on the premise of the 5G operator and thus may ease the latter's data confidentiality or privacy concerns. Furthermore, as shown by our experimental results in Chapter 6, once fine-tuned

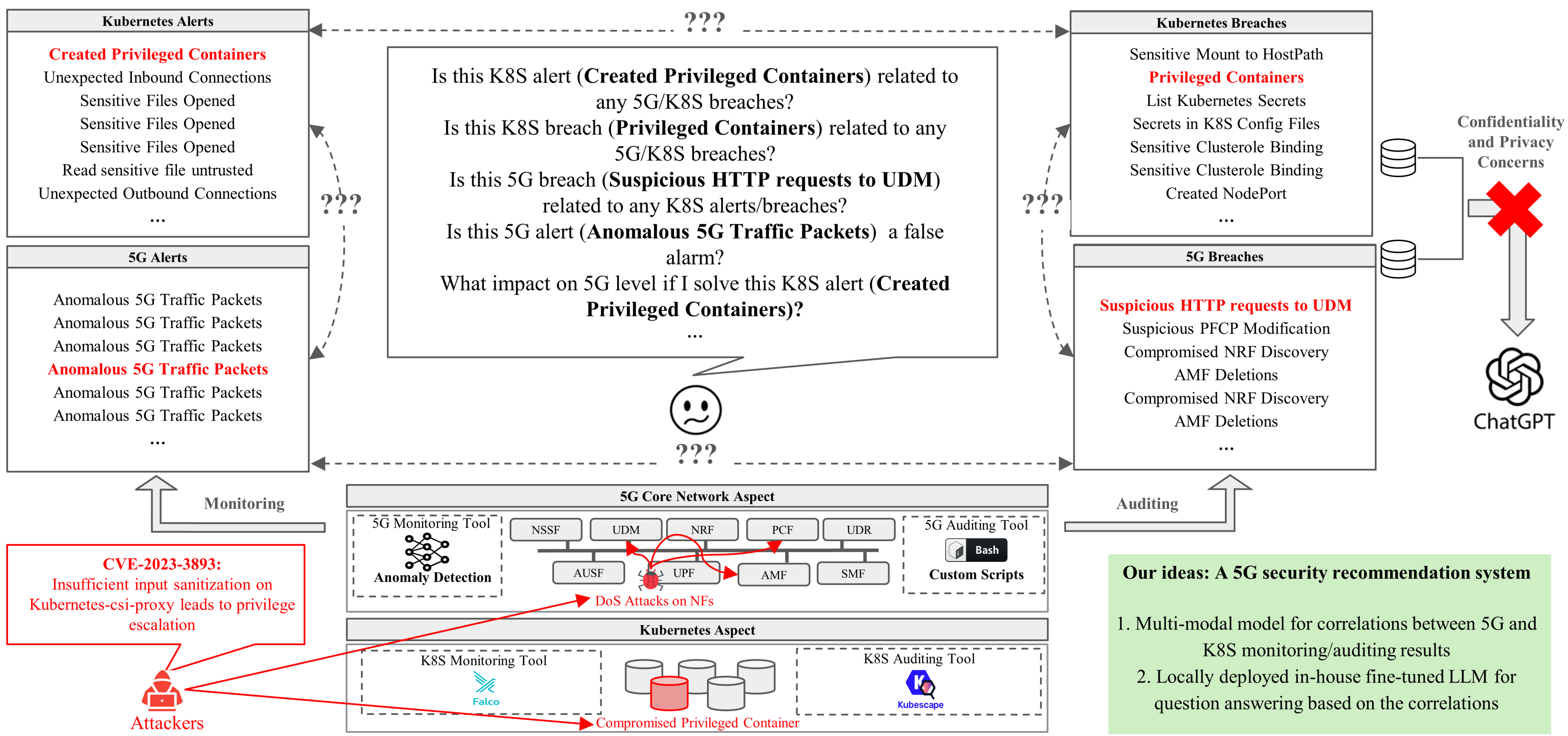


Figure 1: Motivating example

with knowledge specific to event correlations in the 5G deployment, our LLM can provide more meaningful and coherent answers than a general-purpose LLM (e.g., ChatGPT) does. In summary, the main contributions of this thesis are as follows.

- We propose the novel solution of applying multi-modal learning for correlating events between different abstraction levels (service and virtualization) and different security tools (monitoring and auditing) of a cloud-native 5G deployment to a comprehensive view of potential security threats.

- We design a 5G security recommendation system, 5GSecRec, based on an on-premise LLM fine-tuned with the learned knowledge about event correlations (as contextual information), such that security analysts may directly query 5GSecRec with various security-related questions expressed in a natural language about this specific 5G deployment.

- We implement 5GSecRec based on a containerized 5G testbed using free5GC and Kubernetes, and we evaluate its effectiveness and efficiency through experiments. Our results show that 5GSecRec can provide accurate results (e.g., 89.5% accuracy for correlating alerts using multi-modal learning) with a reasonable training time (e.g., 71 minutes for training over 90,000 events with 15 million tokens).

1.5 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 reviews the existing works in the related domain, and compares and contrasts them with our work. Chapter 3 provides the necessary background knowledge to understand 5GSecRec and defines the threat model and its scopes. Chapter 4 describes the methodology and working principles of 5GSecRec. Chapter 5 describes how 5GSecRec was implemented. Chapter 6 presents experimental results and performance evaluation of 5GSecRec. Chapter 7 describes other contributions to this thesis. Chapter 8 concludes the thesis with the potential future research directions and limitations.

Chapter 2

RELATED WORK

To provide a comprehensive understanding of the subject matter, this chapter discusses the related literature related to our work. The latter falls into (1) Cloud-native 5G and its requirements; (2) Alert correlation; (3) Large Language Models (LLMs) question-answering approaches; and (4) Recommendation systems in security.

2.1 Cloud-Native 5G

Transitioning to 5G necessitates the use of cloud-native technologies, which consist of intricate service-oriented cloud architecture, as well as cloud-native functions and applications [20]. This infrastructure and service layers [21] allow various suppliers to create open-source software by combining a variety of possible combinations within and between the layers [22]. Considering different formats of the data, and volume of the generated data, service providers must also figure out ways to reduce operating and capital expenditure expenses while increasing efficiency.

According to Intel, a cloud-native 5G core must comply with customer service level agreements for performance, as well as meet regulatory, privacy, monitoring, security, and auditing standards. All of this must be achieved within a limited power and space capacity while ensuring a satisfactory return on investment [23].

5G network vulnerabilities include hardware, firmware, and software weaknesses, as well as issues with signaling and control plane protocols, containers, and Kubernetes. This is a call to a comprehensive, multi-layered approach required to assess network security and address all service aspects and internal components [24].

2.2 Multi-Aspect Alert Correlations

Some works focus on alert correlation across different aspects. Inam et al. [25] summarizes the use of alert correlations in the auditing investigation process, where they divide existing works into similarity and causality-based correlation while mentioning data sources that show gaps in correlating data from multiple sources. [26] also classifies the correlation approaches into two distinct groups: single and multiple, depending on the data sources used.

Elsoush et al. [10] introduced a framework that streamlines alerts by efficiently filtering irrelevant and false alerts of the intrusion detection system (IDS) by using filter-based correlation and assigning priority to each alert. [11] also offers a general correlation model for IDS that considers all components and shows effectiveness based on different datasets. [27, 28] correlates an alert with an attack graph-based approach to minimize memory usage and predict future alerts and missing alerts. [12] presents an IDS system that uses their proposed ADeLe language to define the correlation properties that correlate either events in the analyzer or alerts in the manager using finite state automata. However, the data source is limited to only a single aspect (e.g., network), therefore it may not be capable of handling data from multiple sources or tools. Additionally, many parts of the solution depend on one or more knowledge bases (e.g., a network information database, CVE, etc.), which makes maintenance difficult and raises security and privacy concerns. Yu et al. [29] propose a framework for alert event reduction by correlating events using neural network and data mining techniques, with simulated data used to demonstrate multiple correlations. There are some studies on alert correlation in software vulnerability detection. By combining different Static Analysis Tools (SATs), [30, 31] enhance software vulnerability detection and lower false alarms through the use of machine learning. While [32] combines various dynamic and static tools with varying degrees of criticality to improve the efficacy of security vulnerability detection. However, these approaches analyze data from only a single aspect (e.g., infrastructure) and are capable of classifying vulnerabilities or non-vulnerabilities, but they fail to address dynamic environments like service-based architectures (e.g., 5G).

Attribute-based alert correlation and attack graphs are used by [33] to reconstruct APTs in large networks. Another attack correlation and scenario reconstruction method are illustrated in [34] through the utilization of an abnormal states relationship graph, graph aggregation, and clustering. Both works make use of an existing IDS dataset that, in reality, only encompasses data from a single perspective (e.g., network). By employing correlation among suspicious information flows, [35] suggests an APT detection method that generates a graph representing the actions of the attacker. Nine real-time attack scenarios were utilized to assess the solution. [36] correlates Endpoint Detection and Response (EDR) alert threats with Tactical Provenance Graphs (TPGs) for multi-stage attack investigation. SIRAJ [37] also combines the detection results from different security tools, such as anti-malware engines, through the application of self-supervised learning and pre-training of an embedding model. Furthermore, [35, 36] also mention that correlation is also used in industry, such as security information and event management (SIEM) tools [38, 39] that use statistical and rule-based approaches to correlate and combine alerts where the correlation is determined by the similarity of events within a specified time interval. However there is no causal relationship between them, and there are no automated suggestions about what caused the alert. Our approach can be used to strengthen decision support in such technologies by capturing the rationale behind potential attacks in an automated manner. This is achieved by going beyond the observability line of such tools and providing recommendations based on contextual information gathered from different aspects. To have good event correlation decision support, the Question-answering capabilities of LLMs need to be investigated, which is introduced in the sequel.

2.3 Multi-Modal and Self-supervised Learning

The combination of multi-modal and self-supervised learning techniques provides a more holistic view of cyber security scope. By integrating diverse data sources and leveraging unsupervised learning methods, security solutions can adapt and learn from different data types, thus, enhancing the ability to gain insights on potential threats, and enabling better analysis for cyber security experts. In the prevailing of these facts, we studied this field from two different perspectives, namely, multi-modal and self-learning techniques in different scopes, and specific use cases related to cyber security.

Multi-model with self-supervised learning [40] has been extensively used in diverse fields, varying from social networks [41–43], medical field [44], emotion and speech recognition [45], media description, retrieval, and generation [46, 47]. The different works pinpoint diverse challenges related to the nature of data, in the name of representation, translation, alignment, fusion, and co-learning. In our use case, we emphasize on events-based data representation and alignment extracted from different logs and auditing results in 5G standalone architecture. The main motivation is to capture the context through the textual nature of log data as well as auditing results, thus, a need to align (sequencing) events and represent them as tokens for a better contextual prediction.

To cyber security, multi-modal deep learning has been employed in some use cases spanning malware detection, cyber threat intelligence, online illicit activities detection, detection of deep fakes, software security as well as network-based detection. For the sake of illustration, Android malware detection using diverse feature types is introduced in [48]. In [49], authors used a multi-modal classification method to identify the dark web's illicit activities. In [50], authors defined an approach to safeguard videos in smart cities against face-faking attacks using deep learning methodologies [50]. In [51], Zhang et al. introduced EX-Action, a framework dedicated to supporting cyber threat intelligence through the extraction of threat actions out of complex and unstructured reports. Software security sees the potential of multi-modal learning with the introduction of the MVDSC-C dataset with a focus on vulnerability detection [52]. In [53], He et al. proposed a multi-modal sequential intrusion detection approach based on packet, traffic, and generic features. The approach performed well on specific binary and multiple classification tasks. In this work, we use a multi-modal approach to capture contextual correlation between different outcomes out of cyber-security tools. In the sequel, we introduce different works related to the correlation.

2.4 Question-Answering by LLMs

Recent work explores the capabilities of LLMs for recommendation systems based on question-answering (QA), especially in domain-specific contexts.

Garza et al. [54] use LLM (GPT-3.5) to examine the understanding of threat behavior in MITRE ATT&CK through the exploration of various input prompts that generate questions and evaluate the quantity of question/answer pairs accuracy with a subject matter expert's opinion. It has been demonstrated that prompts designed with context yield the highest response accuracy on the GPT-3.5, although the LLM level of understanding is still unclear. This study [55] aims to evaluate the efficacy of three language models (OpenAI ChatGPT, Google Bard, and Microsoft Bing) in providing accurate responses to professional certification inquiries and successfully resolving capture-the-flag (CTF) challenges. Additionally, they demonstrate that it is relatively easy to circumvent and undermine the ethical safeguards of LLMs by utilizing jailbreak prompts. Finally, authors in [56] use LLM in network log processing and demonstrate LLM's effective log data parsing capabilities. Nonetheless, LLM was unable to accurately detect future events from the log data or to properly summarize or analyze the log data due to the randomness of the data. One effort [57] tackled the absence of adequate AIoT QA datasets by crafting an AIoT corpus to further pre-trained models like Roberta and BERT, achieving notable enhancements in AIoT QA tasks. Another study [58] introduced LeanContext, a method to curtail the cost of domain-specific LLM responses by judiciously extracting key contextual sentences, showcasing its efficacy against diverse context reduction strategies. A study augmented an LLM with medical textbooks, marking a significant uptick in the professionalism and accuracy of responses for medical QA [59]. Investigations into the ability of LLMs to respond to programming questions laden with source code by fine-tuning them achieved leading results on standard benchmarks [60]. Remarkably, even without task-specific tuning, certain LLMs house relational knowledge that rivals traditional NLP methods and excels in open-domain QA endeavors [13]. Lastly, research illuminated the adaptability of the retriever-reader paradigm, particularly in the Telecom sector, via synthetic data generation, and discerned that sparse retrievers can occasionally overshadow dense counterparts [14].

Recent studies have also concentrated on utilizing LLMs in the telecommunications sector. [61] proposed a framework for identifying technical documents (text classification) in the 3rd Generation Partnership Project (3GPP) (2021-2023) using pre-trained LLMs (BERT [62], RoBERTa [63], and GPT-2 [64]) that have been fine-tuned using 3GPP data (2009-2019). Authors in [65] also use LLM (specifically, BERT) for the question-answering (QA) task in the telecommunication domain.

A LLM based foundation model for 6G communications named NetGPT (Network Generative Pre-trained Transformer) [66] highlights security and privacy concerns (such as backdoor installation and user data privacy) and data governance concerns (such as data proprietary rights, formats, qualities, etc.). Besides data security and privacy, [67] describes other challenges to the telecom industry's adoption of LLM. For instance, the nature and characteristics of telecom data differ from those of common data modalities (e.g., image, text, video), which complicates the process of fine-tuning available LLM with these data for a particular task. Whereas [68] emphasizes on LLMs' parameter adjustment sensitivity, output inconsistency, and computation complexity as telecommunications implementation challenges.

2.5 Recommendation System in Security

Surveys on recommendation systems (RS) in cyber security [69, 70] provide a summary of the most recent work done in the field of cyber security recommendations and show that not many works have been done in this area. RS is predominantly utilized to predict the next probable network attack and to suggest measures to mitigate it. Using collaborative filtering and K nearest neighbor algorithms, Polatidis et al. [71, 72] propose RS to recommend the next probable attack or breach in the network. Soldo et al. [73] use RS for predicting the origins of future attacks based on previous behavior. McDonnell et al. [74] developed CyberBERT, a system based on BERT [62], for identifying malware and predicting the next click. Sayan et al. [75] employ RS to identify potential threats, forecast forthcoming attacks, and propose measures to prevent them. Whereas, Lyons [76] suggests an RS that offers potential courses of action for addressing network intrusions in a specific context. Franco et al. [77] also developed an RS that suggests the most appropriate preventive steps to be implemented during a cyber attack and Sula et al. [78] have suggested utilizing a recommender system to aid in the prevention of Distributed Denial of Service (DDoS) attacks. These studies either concentrated on particular forms of attack or were deficient in terms of data privacy and confidentiality. While certain studies [79, 80] have aimed to enhance the privacy and confidentiality of user data in recommendation systems, they have not provided sufficient detail regarding the utilization of recommendation systems in the field of security.

According to the administrator's preferences and ratings, Esposte et al. [81] provide a recommender system designed to filter and sort alerts arriving from multiple external sources. SecureFalcon [82] utilizes LLM to distinguish between vulnerable and non-susceptible C code samples and to offer recommendations for repairing and recovering vulnerable C code. Ferrag [83] employs the BERT [84] model to detect potential threats and subsequently utilizes FalconLLM [85] to suggest appropriate measures based on incident response and recovery. Nevertheless, both of these works are restricted to specific categories of data.

Chapter 3

PRELIMINARIES

This chapter provides preliminaries, including background information on the cloud-native 5G core network, multi-modal learning, and Large Language Models (LLMs), as well as the threat model.

3.1 Background

3.1.1 Cloud-Native 5G Core Network

At the heart of the fifth-generation mobile network, the 5G Core Network (5GC) employs a service-based architecture (SBA), facilitating communication between various network functions through standardized interfaces [7]. There is a growing shift towards hosting 5GC on cloud-native platforms, where its functions are virtualized, encapsulated in containers, and seamlessly orchestrated. As shown in Figure 2, a cloud-native 5G core network contains components from two levels: (i) The service level includes the 5G network functions, such as the UDM for data management, AMF for user access, UPF for packet direction, Unified Data Management (UDM) for data management, Authentication Server Function (AUSF) for authentication, and Network Slice Selection Function (NSSF) for network selection, Unified Data Management (UDM), Unified Data Repository (UDR) (Table 1 lists common acronyms used in this paper). User Equipment (UE), including smartphones and IoT devices, communicates wirelessly with the core network through the Radio Access Network (RAN), which consists of base stations called gNodeBs (gNBs). These gNBs

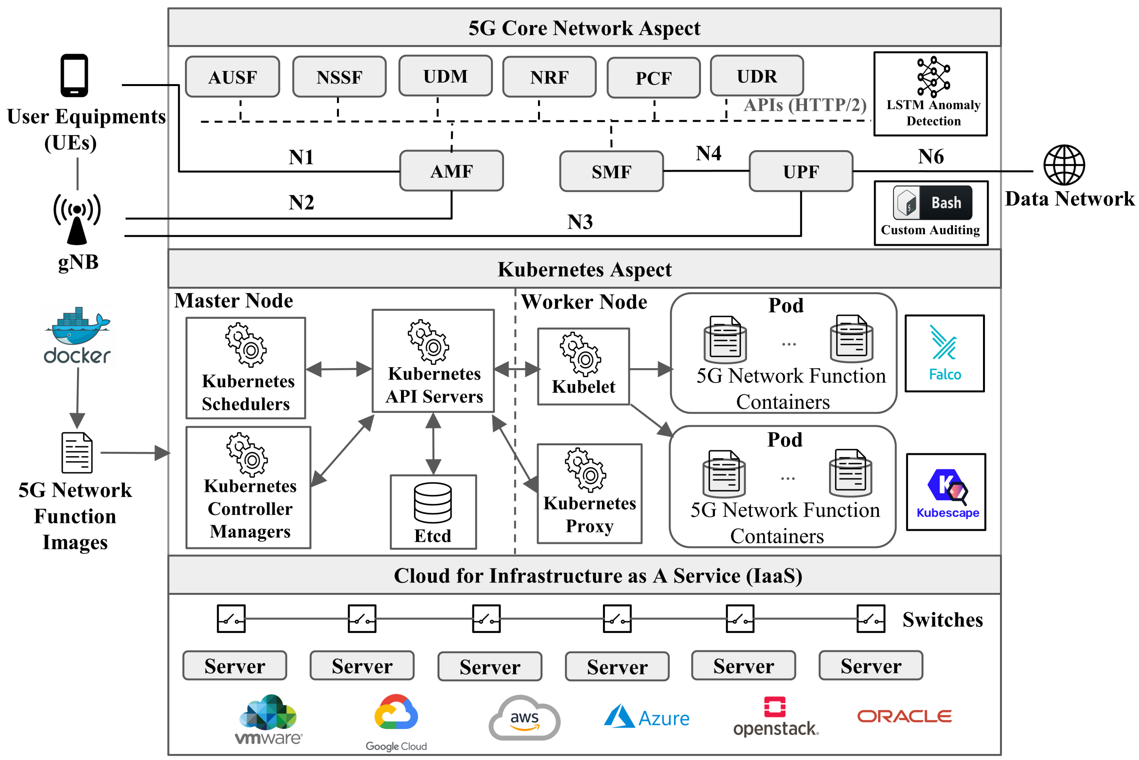


Figure 2: Architecture of cloud-native 5G core network

connect UEs to the service layer, initiating the flow of data and signaling. AMF manages device registration, authentication, and session management. Simultaneously, the SMF is responsible for establishing and overseeing user data sessions. UPF handles the actual user data traffic, ensuring low latency and high data throughput. AUSF plays a vital role in authenticating and authorizing UEs, ensuring that only authorized devices can access the network. NSSF selects the appropriate network slice based on service requirements and user profiles. PCF enforces policies related to traffic management and resource allocation, optimizing network performance. UDM stores and manages user-related data, while UDR stores and manages network configuration information for UEs. These network functions communicate through standardized interfaces and APIs, enabling seamless interaction. Control plane functions handle signaling and network management, while user plane functions manage user data traffic. Key interfaces like N1 (UE to gNodeB), N2 (AMF to UPF), N3 (SMF to UPF), N4 (SMF to PCF), and N6 (UPF to data networks) facilitate communication between specific functions, ensuring efficient service delivery (ii) The virtualization level includes the virtual resources (e.g., containers, virtual machines, and virtual switches) used to host the 5G network based on a cloud infrastructure. A widely adopted container orchestrator at this level is Kubernetes, which supports automated deployment, scaling, and operation of containers across many hosts, with numerous benefits such as auto-scaling, self-healing, service discovery, and load balancing [6].

3.1.2 Multi-Modal Learning

Multi-modal learning in the context of machine learning is an approach that processes and correlates information from multiple types of data (modalities), e.g., text, image, audio, and video. The main motivation for multi-modal learning comes from the fact that real-life objects are usually described in different modalities, e.g., adding a caption to an image, or conversely using an image to describe textual information. Multi-modal learning aims at obtaining a unified model to jointly represent information from different modalities such that the model can capture the correlation structure and apply it for prediction (i.e., recover missing modalities given observed ones). For instance, multi-modal learning has been applied to visual-based question-answering tasks [46, 86, 87], fusion of information from multiple sensors for semantic perceptions in autonomous driving [88–90], and the visual analysis and reasoning of medical images [91, 92].

Figure 3 presents a general example of a multi-modal learning approach employed to train a classifier. Three different types of data—texts, images, and audio are used as inputs. Each data type is processed by its corresponding encoder to extract feature embeddings. These embeddings are high-dimensional vectors that capture the contextual information within each data type. Subsequently, the feature embeddings from the three modalities are concatenated to form a unified feature embedding. This concatenated embedding encapsulates the combined contextual information of all three data types. This composite embedding is then fed into a classification neural network, which is trained to learn the relationships between classes and the combined contextual information. In this work, we regard alerts and breaches coming from different levels of the 5G deployment as different modalities (of the same attack) and apply multi-modal learning to identify the correlations.

3.1.3 Large Language Models (LLMs)

Large Language Models (LLMs) usually refer to deep learning models that have been pre-trained with a large amount of data using a transformer (a deep neural network architecture that utilizes self-attention mechanisms) [93]. Today's LLMs exhibit state-of-the-art performance across various natural language processing (NLP) tasks, such as sentiment analysis, translation, and question-answering. LLMs can be broadly classified as (i) encoder-only models (e.g., Bidirectional Encoder Representations from Transformers (BERT) [62]) mostly for text classification, (ii) decoder-only models (e.g., Generative Pretrained Transformer (GPT) [64]) mostly for generating contextual and coherent text based on given prompts, and (iii) encoder-decoder models (e.g., Text-to-Text Transfer Transformer (T5) [94]) for both understanding and generating text. The details of some typical models are shown as follows.

- *BERT [62].*It is solely based on the encoder part of the Transformer model. This unique focus on the encoder, without incorporating a decoder, is central to BERT's ability to comprehend text context and meaning, rather than producing text. Each encoder in the Transformer model includes a self-attention mechanism and a feed-forward neural network. BERT employs several of these encoders in a stacked configuration. The BERT-Base version contains 12 encoders, while the BERT-Large includes 24. Such an arrangement allows the model to

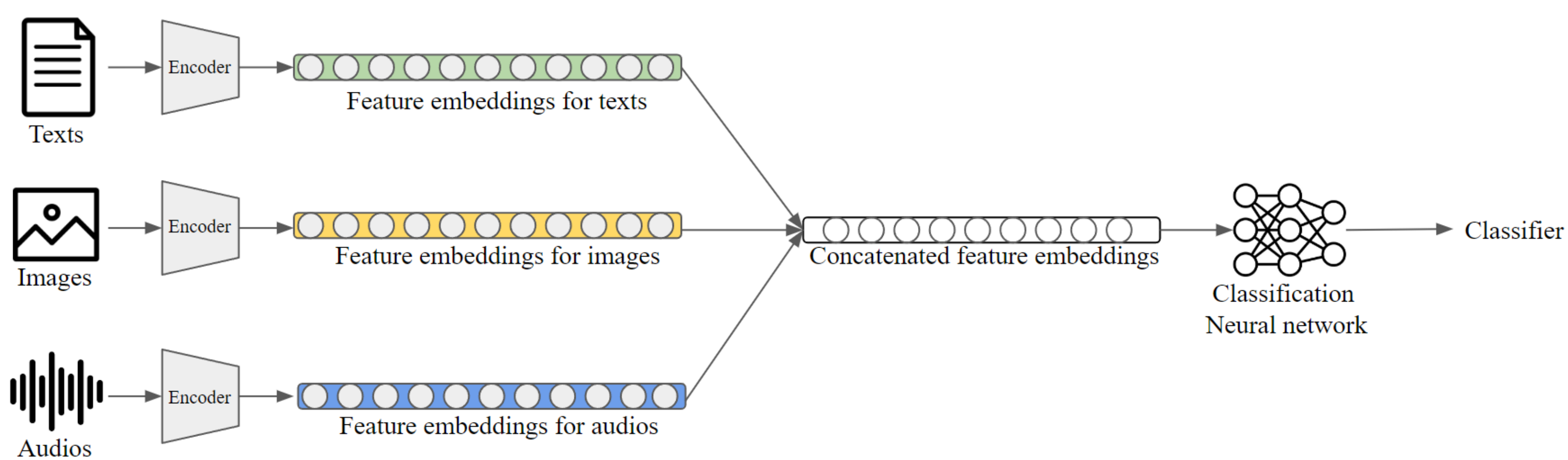


Figure 3: A general example of the multi-modal learning approach

interpret and analyze text at different levels of complexity. One of the standout features of these encoders is the self-attention mechanism. This enables BERT to evaluate each part of the input text, determining how each word relates to and influences others within a sentence. This capability is vital for grasping the subtle context and meaning of words about their surrounding text. BERT is trained in two primary phases: pre-training and fine-tuning. In the pre-training stage, BERT undergoes training on a large text corpus (3.3 billion words from sources like Wikipedia and Google's BooksCorpus) without specific supervision. The training consists of two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, BERT learns to predict words hidden in a sentence, using their context for guidance. In NSP, BERT learns to discern if one sentence logically follows another, enhancing its grasp of language structure and context. The fine-tuning stage involves adapting BERT for particular NLP tasks. This is achieved by adding a few task-specific layers to the pre-trained model and training it on a smaller, task-focused dataset. During this stage, BERT fine-tunes its parameters to excel in specific tasks such as question answering, sentiment analysis, or named entity recognition.

- *GPT [64].* The development of Generative Pre-trained Transformer (GPT) models began with the debut of the original GPT by OpenAI [16]. This initial model set the stage for subsequent versions, each more advanced than the last. Progressing from GPT-2 through to the latest, GPT-4, these models have seen significant enhancements in both their size and their ability to handle complex tasks. In contrast to models like BERT, which are based solely on an encoder mechanism, GPT models uniquely utilize only the decoder component of the Transformer architecture. This specific focus equips them with a natural proficiency in generating text. The decoder-centric design of GPT models renders them particularly effective for text-generation tasks. This includes creative writing, producing content, and crafting responses that closely mimic human conversation in chatbots. Additionally, this architecture makes GPT models apt for scenarios that demand sequential decision-making or output generation, as seen in gaming or interactive storytelling.

- *T5 [94].* It has the encoder-decoder architecture which is commonly designed for machine translation. Unlike other models that are tailored for specific tasks, T5 distinguishes itself by applying this framework across a broad spectrum of natural language processing (NLP) tasks such as question-answering tasks, summarization, and text-generation tasks through a text-to-text method. In this setup, the encoder part of the model first interprets the input text, grasping its context and subtle meanings. Following this, the decoder section comes into play, using the processed information from the encoder to produce the output text. This dual-step approach equips T5 with the ability to efficiently navigate and interpret complex linguistic relationships.

In this work, we leverage encoder-decoder LLMs fine-tuned with the knowledge about correlations for automatically answering security analysts' questions.

3.2 Threat Model

The in-scope threats of this work include any attacks on a cloud-native 5G deployment that can be identified using existing security monitoring and auditing tools. Such attacks may exploit vulnerabilities, implementation flaws, or misconfigurations at either the service level or virtualization level (or both) of the deployment. The out-of-scope threats include zero-day attacks or unknown vulnerabilities/flaws that cannot be identified using monitoring or auditing tools and attacks that can tamper with the integrity of such tools (including their inputs and results) or 5GSecRec itself. We assume the availability of necessary training data for the machine learning models behind 5GSecRec to be periodically re-trained to capture system dynamics, and such training is not subject to adversarial attacks (e.g., leveraging existing approaches [95–97]). We also assume loose synchronization of events across different aspects of the 5G deployment such that machine learning can properly identify their correlations. Finally, 5GSecRec is designed as a security recommendation system to ease the job of security analysts in understanding attacks. As such, it is neither intended to replace human experts nor designed as a standalone attack detection, auditing/verification, or prevention system.

Term	Description	Term	Description
3GPP	3rd Generation Partnership Project	5GC	5G Core
AUSF	5G Authentication Server Function	CN	Core Network
K8S	Kubernetes	LLM	Large Language Model
AMF	Access and Mobility Management Function	NF	Network Functions
NRF	5G Network Repository Function	NSSF	5G Network Slice Selection Function
SMF	5G Session Management Function	UDM	5G Unified Data Management
UDM	Unified Data Management	UDR	Unified Data Repository
UE	User Equipments	UPF	5G User Plane Function
RAN	Radio Access Network	PCF	Policy Control Function

Table 1: Acronyms used in this thesis

Chapter 4

METHODOLOGY

This chapter first provides a high-level overview of the 5GSecRec methodology and then details its two stages.

4.1 Overview

Figure 4 shows an overview of the 5GSecRec approach which contains two main stages as follows.

- The *learning* stage is to learn (a) alert correlation as well as (b) answers to security analysts' customized questions. Specifically, to *learn alert correlation*, 5GSecRec first collects and pre-processes historic results from various 5G security solutions (e.g., Kubescape [9], Falco [8], and anomaly detector based on LSTM [98]) and trains a multi-modal model where the results reported by each security solution are considered as a modality. To *learn answers to customized questions from security analysts*, it first prompts powerful LLMs (e.g., GPT-4 [16]) to generate high-quality training data by obtaining experts' knowledge of historical results, and then fine-tune that LLM model (noted as "QA-LLM") for question-answering tasks.
- The *running* stage is to discover alert correlations among new alert results from those 5G security solutions as well as to respond to specific security analysts' questions about those

alerts. Specifically, to *discover alert correlations*, 5GSecRec utilizes the trained "multi-modal" model to deduce potential alert correlations and express them in natural language. To *respond to analysts' queries*, it first determines the most appropriate question-answering task and its relevant prompt template by using sentiment analysis, then combines with the current alert correlation result to finalize the prompt selection for this specific task (following the effective prompt engineering principles [99]), and finally generates recommendations using the selected prompt into the "QA-LLM" model.

4.2 Learning Stage: Learning Correlations among Security Events

In this stage, 5GSecRec learns correlations among security events (alerts or breaches) generated by both auditing and monitoring solutions deployed at both 5G and Kubernetes levels.

Challenges. Attacks on a cloud-native 5G core network can cause security appliances, such as auditing and monitoring tools of Kubernetes (K8S) and 5G core network, to report a substantial volume of alerts and breaches within a short period of time. Therefore, manually identifying the correlations between these alerts and breaches to understand their potential dependencies and see the big picture is usually non-trivial. Additionally, this task becomes more challenging as alerts or breaches from the K8S and 5G levels typically do not share a common data source, and the corresponding security tools may not be compatible with each other (e.g., different scopes, different criteria for triggering alerts and breaches, and different formats of their results). Furthermore, 5G security currently lacks large-scale publicly available labeled datasets which are necessary for training popular machine learning models. In the following, we describe how we overcome these challenges.

Our solution. To streamline this process, we leverage multi-modal (to understand the correlation among security results from various security solutions) and self-supervised learning techniques (to avoid the need for very large-scale datasets). Specifically, for multi-modal learning, we aggregate security results from four aspects: Kubernetes monitoring, Kubernetes auditing, 5G monitoring, and 5G auditing, while designating each aspect as a unique 'modality' due to the complementary nature of those four aspects. Within each modality, results are represented as *tokens*, and these result

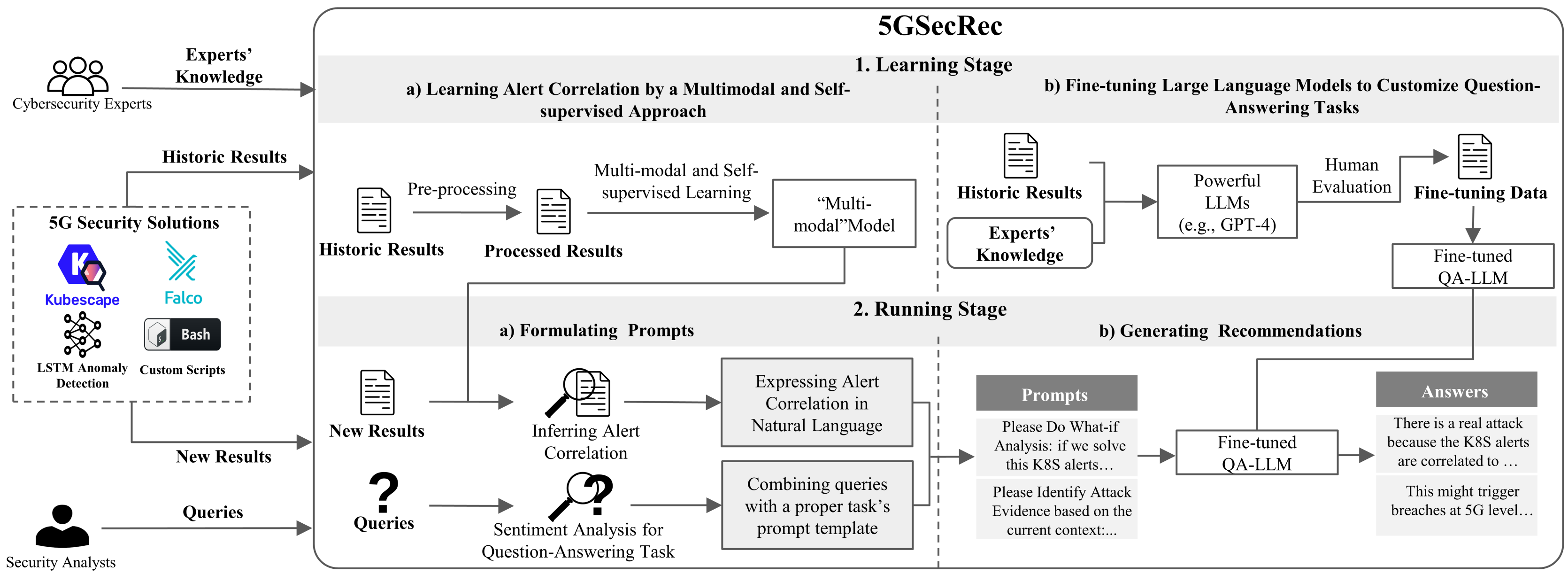


Figure 4: An overview of the 5GSecRec methodology

tokens are organized to form result *sequences*, creating a unique *context*. Then we leverage the contextual understanding capability of LLMs to learn a joint representation of contexts from those sequences. For self-supervised learning, given a set of results with specific details, we selectively and randomly hide (a.k.a. mask) part of the details of each result. The masking could be applied to either the title of a result or all its details. The model is then trained to predict or fill in these masked parts, essentially reconstructing the original, unmasked set of events. The accuracy of the model's reconstruction, measured by a loss value, is used to guide the continual improvement of the model. Consequently, when provided with an incomplete set of data, the model can predict the hidden parts of events and their details. In doing so, the model also learns how the events in the set are contextually related to each other.

In particular, for self-supervised learning, Our solution applies transformer-based models [62, 64, 94, 100]. During training, the models learn the correlations between different events across the four modalities. For example, the models will learn how Falco alerts such as *unusual inbound connections* and *sensitive pod actions*, often occur with Kubernetes breach of *privileged containers*, and how these are linked to certain 5G alerts or breaches. Once the models can leverage the transformer architecture to understand the context created by these combinations, they can be applied to predict events from other modalities based on a given event and the context.

More formally, let $E^i = [e^i_1, e^i_2, e^i_3, \ldots]$ be the sequence of events from modality i, and let S^i be a special token that represents the separator for modality i. The concatenated sequence C with separators can be represented in Equation 1, where $[S^i, E^i]$ represents the concatenation of the separator and the event sequence for modality i.

$$C = [S^1, E^1, S^2, E^2, S^3, E^3, \ldots] = [S^1, e^1_1, e^1_2, \ldots, S^2, e^2_1, e^2_2, \ldots] \tag{1}$$

The reconstruction loss can be represented as the difference between the predicted events and the original events as follows:

$$\text{Reconstruction Loss} = \sum_{j \in J} L(\hat{e}_j, e_j) \tag{2}$$

where:

- $\Sigma_{j \in J}$ denotes the sum over the set of indices J, where the events have been masked.
- $L(\hat{e}_j, e_j)$ represents the loss function comparing the predicted event $\hat{e}_j$ at position j with the original event e_j.
- L is a specific loss function such as Mean Squared Error (MSE) or Cross-Entropy.
- $\hat{e}_j$ is the model's prediction for the masked event at position j.
- e_j is the actual (original) event at position j.

Example. Figure 5 shows an example (with the security results from Falco and Kubescape corresponding to the attack scenario described in our motivating example) of our two learning correlation steps. In (1a), from Falco alerts and Kubescape breaches, 5GSecRec keeps the information that is common to multiple results, the titles of alerts (*unexpected inbound connections* and *exec into pod*) and breaches (*privileged containers* and *sensitive mounts*), and removes other information, such as timestamps and IDs, which typically vary in each result, to assist the learning in the following step. In (1b), we group the filtered results from both Falco and Kubescape, order results from each group, and combine them into a sequence, using markers, *[Falco]* and *[Kubescape]*, for separation. In (2), 5GSecRec converts those sequences into text embedding as mentioned above.

5GSecRec applies transformer-based models for the reconstruction. As mentioned in Section 3.1, transformer-based models such as BERT, GPT, and T5 leverage transformer architecture to achieve state-of-the-art performance in a wide range of NLP tasks (e.g., text generation, question answering, etc.). This is because the transformer architecture can process effectively long-range dependencies, where the model maintains accuracy even across extensive sequences. Additionally, the scalability of transformers is a key advantage. As these models grow in size, they continue to improve, making the most of larger datasets and enhanced computational resources. This growth enables transformers to develop a deeper understanding of nuanced language patterns and subtleties.

Its core strength lies in its ability to interpret contextual relationships. This feature is not limited to words in text; it's equally effective in analyzing visual data, like the relationships between pixels in an image. The way transformers approach learning is also noteworthy. By employing strategies like pre-training and fine-tuning, they adeptly capture and replicate complex language structures and patterns. Figure 6 shows a visual representation of the transformer architecture. The transformer

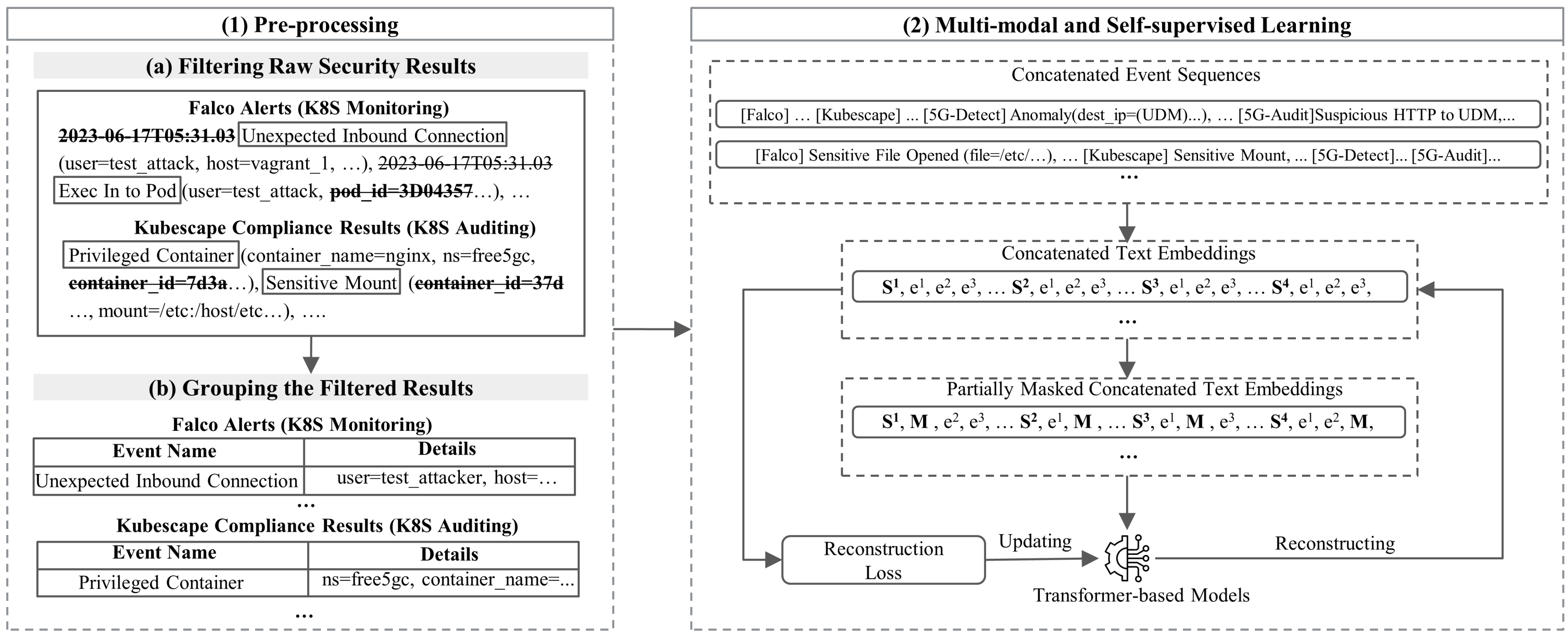


Figure 5: An example of learning correlation among various security results

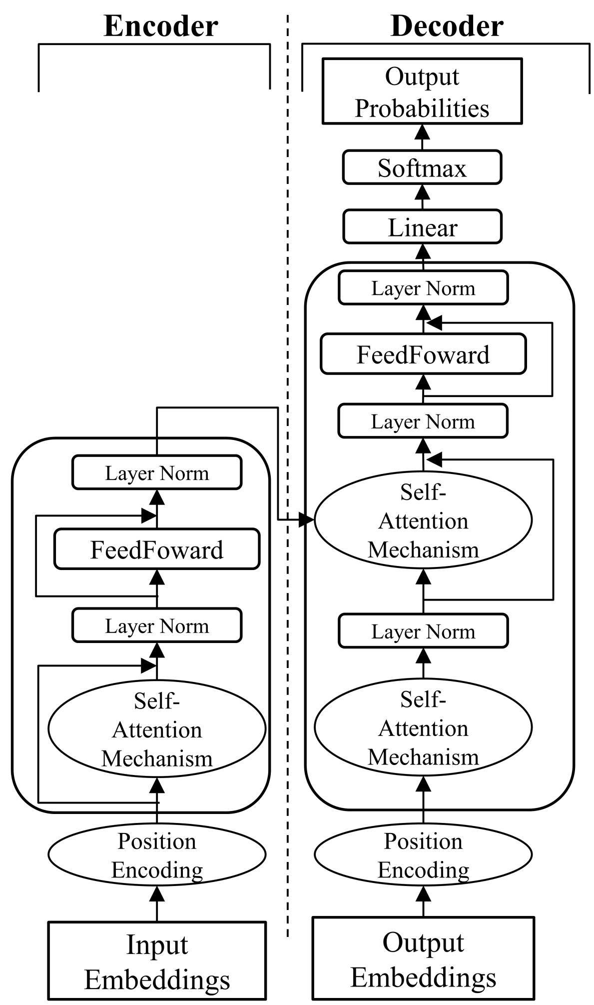


Figure 6: Overview of the transformer architecture

has two primary components: the encoder and the decoder, each comprising several layers that process data in different ways. In the encoder, the input embeddings convert the words or tokens in the input data into vectors. These vectors are numerical representations that encapsulate the semantic and syntactic characteristics of each word. The self-attention mechanism in the encoder then allows each word to interact with every other word in the input, helping the model understand the context and relationship between words. After the self-attention process, each encoder layer applies a feedforward neural network to the data, further refining the understanding of the input. Layer normalization is another critical component applied within each layer of both the encoder and decoder. It normalizes the data across features, which helps in stabilizing the learning process and leads to more efficient training of the model. In the decoder part, which is responsible for generating output, things work a bit differently. The output embeddings convert the previously generated tokens into vectors. The decoder then uses a masked self-attention mechanism, which is crucial as it prevents the model from seeing future tokens in the sequence. This ensures that the output is generated one token at a time in a forward direction. After processing the data through its layers, the decoder's final step involves a linear layer followed by a softmax function. The linear layer adjusts the size of the decoder's output to match the size of the vocabulary. Then, the softmax function turns these outputs into a probability distribution, indicating the likelihood of each token being the next element in the sequence. The model selects the token with the highest probability as its output, and this output is then fed back into the model as part of the input for generating the next token.

In 5GSecRec, during training, the models learn the correlations between different events across the four modalities. For example, the models will learn how Falco alerts such as *unusual inbound connections* and *sensitive pod actions*, often occur with Kubernetes breach of *privileged containers*, and how these are linked to certain 5G alerts or breaches. Once the models can leverage the transformer architecture to understand the context created by these combinations, they can be applied to predict events from other modalities based on a given event and the context.

4.3 Learning Stage: Fine-tuning Large Language Models (LLMs)

To answer 5G security analysts' customized questions, it is desirable to utilize popular large language models (e.g., GPT-4 [16], and Llama-2 [100]). However, data confidentiality and privacy concerns may prevent a 5G operator from uploading their data to third parties hosting such models and using those general-purpose models without fine-tuning them with domain-specific data may lead to inaccurate or misleading results (as shown by our evaluation results in Chapter 6). To address this challenge, we first generate fine-tuning data with our domain-related question-and-answer tasks spanning a diverse range of desired use cases (e.g., identifying false alarms, gauging attack impacts, and what-if analyses) using popular LLMs (e.g., GPT-4). We then utilize those data to fine-tune an open-source LLM (e.g., Llama-2). We elaborate on these steps as follows.

Generating Fine-Tuning Data by Powerful LLMs. Generating fine-tuning data, typically with question-answer pairs, requires significant human effort. We thus leverage powerful LLMs, like GPT-4, to produce the necessary diverse fine-tuning data at scale. For instance, we provide GPT-4 with historic results and cybersecurity experts' insights elucidating the meaning of the results, the essence of the question-answering tasks, standard question templates, etc. These instruct GPT-4 about the generation criteria for fine-tuning data. Once GPT-4 yields raw fine-tuning data, human intervention is essential to assess, modify, or filter question-answer pairs. The historical results sent to GPT-4 originate from a 5G testbed designed to simulate the production system, which mitigates potential concerns related to confidentiality or privacy (for operators who would be reluctant to send even such simulated data to third-party LLMs like GPT-4, open-source LLMs can be considered as alternatives for this step).

More specifically, we first collect some simulated historical security results from the four aspects (Kubernetes Monitoring, Kubernetes Auditing, 5G Monitoring, and 5G Auditing) of the testbed. We then utilize expert knowledge to construct scenarios that illustrate how certain responses might assist security analysts, given these security results. Additionally, we provide templates of questions that security analysts are likely to pose. Finally, we leverage the powerful LLMs to utilize their pre-trained knowledge for generating high-quality candidate fine-tuning data, which can then be further refined and tailored to better meet our requirements. An important consideration here is adaptability, i.e., as the questions at runtime might differ from those in the training dataset due to varying contexts, the question-answering LLM is considered effectively fine-tuned to recognize the desired response pattern, if it can infer the most probable answers when confronted with slightly different questions (as evaluated through experiments in Chapter 6).

General Guidance for Experts. To effectively guide powerful LLMs in producing high-quality data, experts are advised to adhere to the following:

- *Clarity in Task Definition.* Define the task clearly by detailing its specifics, including the task's purpose, and the context in which it is performed, and providing diverse question-answering samples that encompass various scenarios relevant to the task's domain.
- *Explanation of Insights in Historic Results.* Provide a clear explanation of how security events within the context are indicative of insights, such as potential attacks.
- *Quantity and Quality of the Generated Data.* Indicate the required amount of data and define quality standards. These standards should encompass the accuracy and relevance of the data generated.
- *Human Evaluation.* Utilize trained and diverse reviewers to perform evaluations on fine-tuning data based on accuracy, and relevance.

Example. Figure 7 shows an example of generating fine-tuning data using GPT-4 based on given expert knowledge and historical results. The expert knowledge is obtained by studying the correlation results (from Example 3.1) as follows.

- *Domain Specification.* This includes detailed insight into the correlation results. First, based on its purpose, the task is titled as *Identify Attack Evidence*. Second, the content of the task (e.g., purposes) is described. Third, the expert enlists sample question formats, such as "*Given the Falco alert (created privileged container), are there any correlated alerts or breaches from other aspects to support this alert as a real attack?*" as prompt and "*Yes, this suggests a real attack. The detection of a privileged container's creation by Falco, combined with historical data of suspicious HTTP requests to UDM, indicates remote access and privilege escalation.*" as the desired answer. Finally, the expert provides his/her explanation on how a Falco alert indicting a privileged container creation might lead to a 5G audit breach for suspicious HTTP requests to UDM.

- *Instructions for Generating Fine-tuning Data.* This includes the instructions (e.g., data format and quantity) to LLMs about the training data. First, an expert indicates the data format containing: a *prompt* (for questions and historic results) and *completion* (for the answers with explanation). Second, s/he mentions generating five prompt-completion pairs based on our question format.

Fine-Tuning Open-source LLMs for Question-Answering. This step is to fine-tune a pre-trained LLM such that it can be more specialized for our intended security recommendation tasks. We choose an open-source encoder-decoder LLM such as Llama-2 [100] to facilitate the fine-tuning. Although such LLMs are already trained on vast amounts of general-purpose data (which gives them a broad understanding of natural languages and the ability to generate coherent text), they are not specialized for any particular security task, such as providing security recommendations for 5G networks based on correlated security events. Therefore, we first provide additional context and prompt (following the typical question-answer pair input format of those LLMs) about a specific security task. We then fine-tune the LLMs by training them with such specific data. The fine-tuning process does not fundamentally change the model's core abilities but simply causes the model's parameters to be slightly adjusted such that the model becomes more proficient for this particular security task.

4.4 Running Stage: Inferring Correlations and Formulating Prompts

This is the first step during the runtime stage where 5GSecRec infers correlations among the current security results using the learned multi-modal model, and formulates the prompt to be fed to fine-tuned LLMs for security recommendations.

Inferring Alert Correlations by Multi-Modal Model. This step is to apply the learned multi-modal model for inferring correlations among the events from the four aspects (Kubernetes monitoring, Kubernetes auditing, 5G monitoring, and 5G auditing). The inference of correlations can

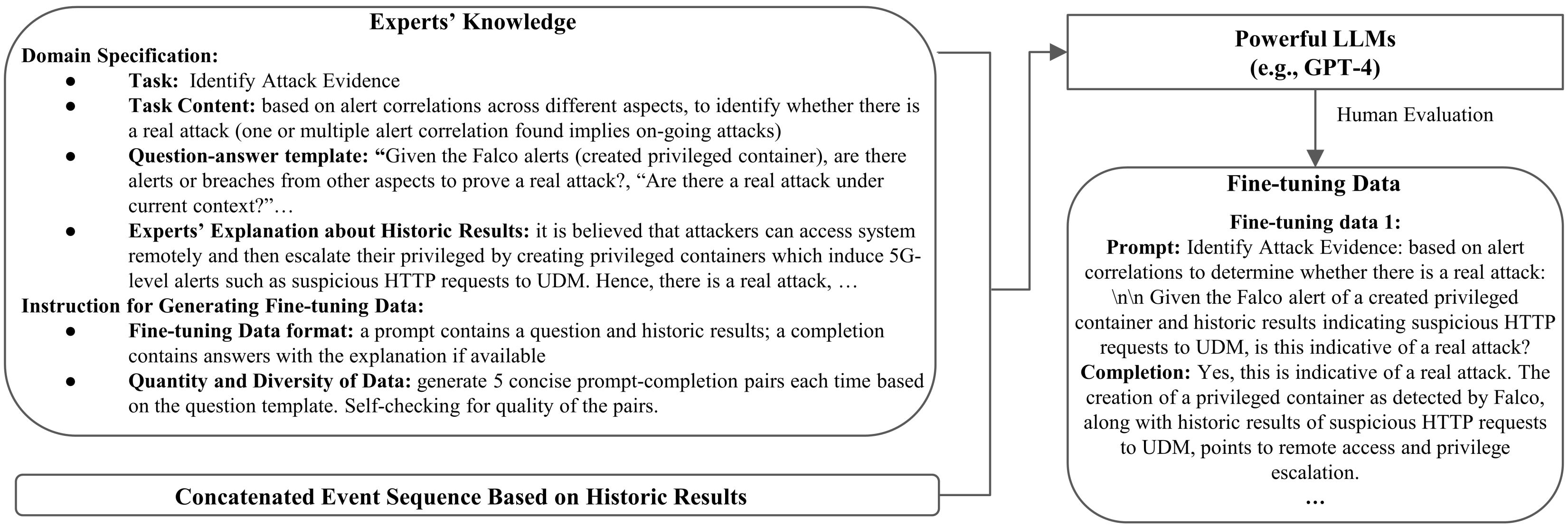


Figure 7: An example of generating fine-tuning data by powerful LLMs

enhance the capability of question-answering LLMs to respond to security analysts' queries with greater accuracy and relevance. Specifically, at runtime, when a security analyst selects some specific events from one or more aspects, these selected events serve as the input for the trained multi-modal model to perform inference of correlations. As the training process has employed masked token prediction for the model to supplement missing event tokens to complete potential event sequences, the model can now make inferences about the probability of potentially correlated events from other aspects based on the given event selected by the security analyst. The inference determines potential sequences of correlated events while assigning each event a probability and ranking the event sequences in descending order based on their respective probabilities. We can decide to select the top n most probable event sequences. If the predicted correlated events occur together in the runtime events, then they are marked as *correlated*, otherwise as *uncorrelated* or *reserved for future*.

Example. Figure 8 illustrates an example to show how 5GSecRec finds correlated security events for a specific set of selected events: *unexpected inbound connection* alert from Falco, *privileged container* and *sensitive mount* breaches from Kubescape, and *anomalous traffic to UDM* from 5G detect. First, 5GSecRec introduces a placeholder token, *[M]* (i.e., masked events), to represent any potential missing events in those sequences. Second, it leverages the trained multi-modal model to infer candidate sequences, filling the *[M]* token with probable events for each aspect. If the multi-modal model predicts that no missing events are needed to replace *[M]*, *[M]* will be replaced with an empty string. Third, it produces two candidate event sequences from the model prediction results: *Candidate 1* including *unexpected outbound connection* (Falco), *clusterrole binding* (Kubescape), and *suspicious HTTP requests to UDM* (5G auditing), and *Candidate 2* with different events. Finally, 5GSecRec marks the events from these candidates that are also found among the runtime events as *correlated events*, and those among the runtime events but not in candidate sequences as *uncorrelated*. The events that are not present among runtime events may be reserved for other purposes such as predictive tasks in prevention strategies.

Algorithm 1 is designed to analyze user-selected events within various aspects (e.g., K8S Monitoring, 5G Auditing) and correlate them with events occurring at runtime. It utilizes separators to format event groups into sequences and then processes them using a multi-modal model. The top

Algorithm 1 InferEventCorrelation(Events, Aspects, Runtime Events, Separators)

Require: $Events$ - Set of user-selected events
Require: $Aspects$ - Set of aspects (K8S Monitoring, K8S Auditing, 5G Monitoring, 5G Auditing)
Require: $RuntimeEvents$ - Events occurring in the system at runtime
Require: $Separators$ - ([Falco], [Kubescape], [5G Audit], [5G Detect])

```
Group Events by Aspects into EventGroups
Format EventGroups into Sequence S using Separators
Process S through the multi-modal model
Let C be the set of candidate sequences from the model
Select top n sequences from C as TopSequences
for each sequence seq in TopSequences do
    if seq ⊆ RuntimeEvents then
        Add seq to CorrelatedEvents
    else
        Add seq to PreventionSequences
    end if
end for
Set UncorrelatedEvents = RuntimeEvents \ CorrelatedEvents
Return (CorrelatedEvents, UncorrelatedEvents)
```

Algorithm 2 ProcessWhatIf(Q)

Require: User's question Q

```
isWhatIf ← AnalyzeSentiment(Q)                ▷ Determine if Q is a what-if analysis
if isWhatIf then
    E ← ExtractTargetEvents(Q)
    S_candidate ← InferEventCorrelation(Seq_new,
        [appropriateAspects], [RuntimeEvents], [Separators])
    for each sequence in S_candidate do
        Check for the existence of events in runtime data
        Classify sequences into correlated or uncorrelated groups
    end for
    return the classified sequences S
end if
```

candidate sequences are selected based on predefined criteria. These sequences are classified as correlated or uncorrelated with runtime events. The resulting sets, "CorrelatedEvents" and "UncorrelatedEvents", are the outcomes. Algorithm 2 evaluates a user's question, determining if it pertains to a what-if analysis. If so, it extracts target events from the question, generates a new sequence, and utilizes the "InferEventCorrelation" algorithm to identify candidate event sequences. These sequences are then checked for correlation with runtime events and classified accordingly.

Formulating Prompts. This step is to formulate a prompt for a specific query from security analysts. We first analyze the sentiment of analysts' queries to identify the type of question-answering tasks (e.g., identifying attack evidence and identifying attack impact). Second, to make sure our QA-LLM (obtained in Section 4.3) correctly understands these queries, we use prompt engineering, as discussed in [99]. This step is crucial to prevent the LLM from giving wrong or irrelevant answers since a well-designed prompt template is known to improve the quality of answers from QA-LLMs. The template is obtained based on the domain specifications introduced in Section 4.3 (which helps to create better training data using advanced LLMs). For each question-answering task, the template describes the task's goal and any special instructions for the QA-LLM. We will provide more details about these templates in Table 2 in the following section.

Example. Figure 9 considers a scenario where a security analyst poses the following queries: *Query 1:* "what are the current attack methods?", *Query 2:* "What are the consequences of those 5G-level alerts?", and *Query 3:* "How often do those Falco alerts occur?". Following sentiment analysis, *Query 1* is associated with the task *Identify Attack Evidence*, and *Query 2* with *Identify Attack Impact*. For *Query 3*, the NLP engine is unable to determine the question-answering task and hence 5GSecRec marks it as *Unknown*, which will be returned for the analyst to rephrase them for more accurate processing.

4.5 Running Stage: Generating Recommendations

This step is to formulate recommendations based on security analysts' queries. Specifically, once the task template is accessed and correlation is inferred, 5GSecRec first combine them to form the final prompt directed to the fine-tuned QA-LLM from Section 4.3. 5GSecRec then leverages

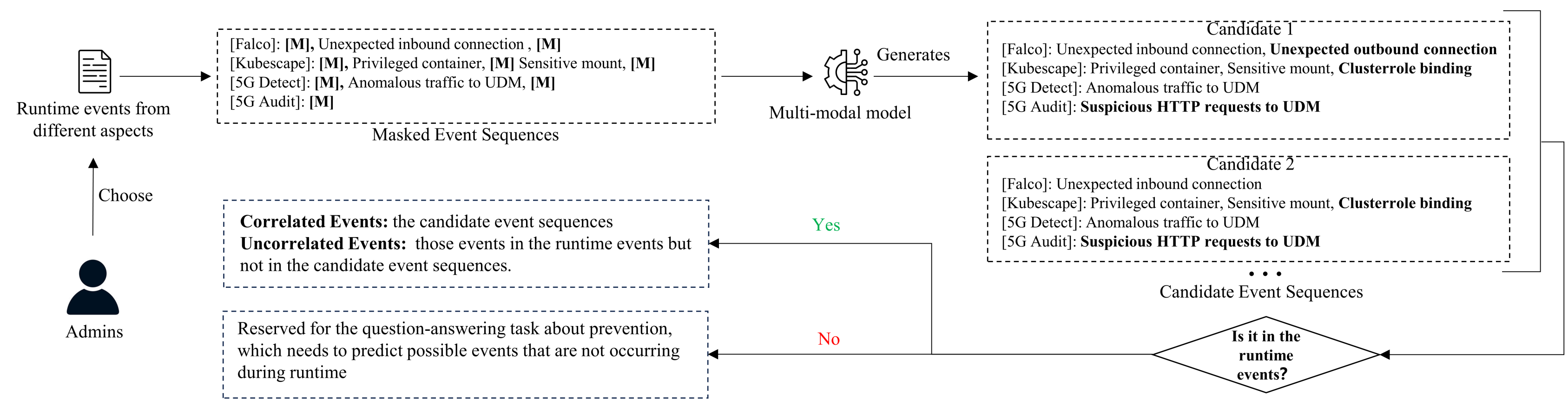


Figure 8: Inferring alert correlations by the multi-modal model

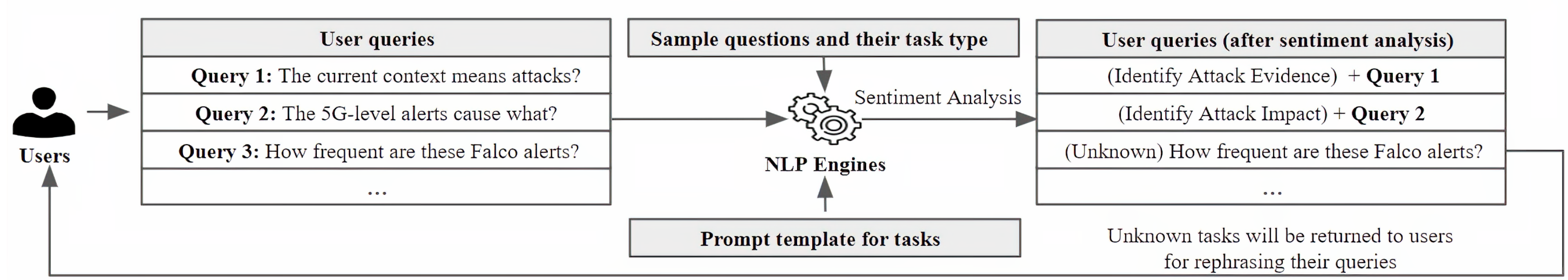


Figure 9: An example of identifying task type

the acquired question-answering capabilities of QA-LLM to formulate recommendations grounded in correlation results to answer analysts' queries. Table 2 lists several representative categories of question-answering tasks that are typically part of a security analyst's workload [101] as follows.

- *Identify Attack Evidence:* This is to determine if there is additional evidence of an attack based on the correlation results. The task involves analyzing alerts, such as Falco alerts or 5G breaches, to identify (the lack of) correlated events from other aspects. The absence of such correlated events might suggest a false alarm, while their presence can trigger further analysis to determine if they provide sufficient evidence for a genuine attack. The typical question template is: "Are there any indications from other data sources that validate the current alerts as evidence of a real attack?".

- *Identify Attack Impact:* This focuses on assessing the consequences of an attack. This task analyzes the impact of specific events based on correlated events, especially those from aspects more interesting to analysts (e.g., 5G-level impact). The typical question template is: "What are the potential consequences of the current alerts on other aspects?".

- *Identify False Alarms:* Unlike the first category that focuses on identifying the actual evidence, this focuses on the mere absence of correlated events. The goal is to identify alerts that lack supporting evidence from other aspects, such as corresponding security breaches, which can trigger further analysis as potential false alarms. The typical template is: "Which recent alerts are likely to be false alarms?".

- *Prevention:* This allows analysts to reason about how to prevent certain events from occurring based on known correlations, e.g., fix a security breach to prevent an imminent attack known to exploit that breach or block certain attacks to prevent a given impact. The typical question template is: "What alerts or breaches should be prevented in the future given the current events?".

- *What-if Analysis:* This is to enable analysts to explore hypothetical scenarios, e.g., to understand the potential benefit of deploying a security tool, or to forecast the potential consequence of current events. The typical question template is: “What possible outcomes might arise if we solve these alerts or breaches, or if we see new alerts or breaches in the current context?”.

The description of each of those tasks can function as a prompt template, guiding the QA-LLM to align its responses with the intended objective of each task (as illustrated in the following example).

Example. Table 4 shows examples on how 5GSecRec responds to two types of question-answering tasks: (i) identifying attack evidence, and (ii) what-if analysis. For the first task, the analyst’s question is “*Given the Falco events, is there evidence from other aspects to prove it is a real attack?*”, which aligns with the *identifying attack evidence* task. In this context, alerts and breaches have been reported from four aspects: Falco, Kubescape, 5G Audit, and 5G Detect. QA-LLM then provides an explanation regarding the logic behind these correlations (as shown in the ‘Answers’ column) as an answer to the query. For the second task, the query is “*What if I resolve the Kubescape breaches (privileged container and sensitive mount) in the current context?*”, which is categorized as a *What-if analysis* task. This task explains the potential correlations that could arise in a hypothetical scenario where the user would resolve the Kubescape breaches (*privileged container* and *sensitive mount*). Similar to the first task, in this context, alerts and breaches are gathered from Falco, Kubescape, 5G Audit, and 5G Detect. The QA-LLM provides an explanation regarding the logic behind these correlations (in the ‘Answers’ column) as an answer to the query.

Question-answering Tasks	**Task Descriptions**	**Question Templates**
Identify Attack Evidence	This task aims to determine the presence of a real attack based on confirmed alert correlations. If no correlated events are found, it may suggest that no known attacks are occurring. However, if correlated events are detected, the task is to interpret how these events contribute to an attack	With the given Falco alerts or breaches, is there evidence from other aspects to prove it is a real attack? Observing these 5G alerts or breaches, is there correlation evidence to prove it is a real attack?
Identify Attack Impact	This task is focused on determining the consequences of various aspects in response to the given input events. It involves analyzing the alert correlations related to the input events. If no alert correlations are found, this may indicate that there are no further consequences. If correlations are present, it is necessary to interpret why the input events have led to certain outcomes.	Observing this Falco alert (unexpected inbound connection), what could be its impact at the 5G level?
Identify False Alarms	The objective of this task is to filter out false alarms. Uncorrelated events are considered likely false alarms, as they do not exhibit patterns similar to those in the training data. The task involves interpreting why correlated events are linked to each other and why uncorrelated events display no such correlations.	Given these alerts/breaches at the K8S level, what could be the possible false alarms? Given these alerts/breaches at the 5G level, what could be the potential false alarms?
Prevention	This task focuses on predicting events that are not occurring in the current runtime but may be correlated with the input events. Identifying these potential events early is crucial for effective prevention. The task requires an understanding of why the input events might be correlated with these potential future events.	What can be done at the K8S level to prevent this 5G attack/alert?
What-if Analysis	This task aims to forecast potential events across various aspects under hypothetical conditions, primarily defined by the presence or absence of certain events. In cases where new alert correlations emerge due to these conditions, it is necessary to explain these correlations. Conversely, if no new alert correlations are detected, the outcome would be a report of no new findings.	What would be the consequences at the 5G level based on the current Falco alerts such as created privileged container; With current K8S breaches from Kubescape, what would be the consequence at the 5G level?

Table 2: Question-answering tasks and their purposes and question templates

Question Answering Task	Prompts	Answers
Identify Attack Evidence	**Task's Prompt Template**: Please identify attack evidence: attacks are based on valid alert correlations. If no alert correlations are found in the current context, this may indicate that there is no attack evidence. If any correlations are found, it means there is a real attack, and it is necessary to interpret why the correlations contribute to a real attack. **Question**: Given the Falco events, is there evidence from other aspects to prove it is a real attack? **Context**: [Falco]: Launch_Privileged_Container (user: test, container:attack-nginx, ns:free5gc, image:nginx), Launch_Sensitive_Mount (user: test, host_path:/var/run/docker.sock, verb: create) [Kubescape]: Privileged_Container (container:attack-nginx, ns:free5gc), Sensitive_mount (mount=/var/run/docker.sock) [5G Audit]: Steal_Data_from_UDM (an HTTP/2 request to UDM from unexpected IPs other than NFs' IPs) [5G Detect]: Suspicious HTTP/2 traffic from an unknown IP (dest: UDM, protocol: HTTP/2)	This evidence supports the likelihood of a real attack, based on the following: Falco alerts have identified the creation of a privileged container, named attack-nginx and using the Nginx image, in the free5gc namespace. Kubescape has also reported a breach, confirming the presence of a privileged container with the same name in this namespace. Typically, attackers can leverage such privileged containers to escalate their privileges and potentially compromise the 5G network. This could involve stealing user data from the Unified Data Management (UDM) via HTTP/2 requests. These requests can be detectable by a 5G anomaly detection tool, which has identified that these HTTP/2 requests are indeed reaching the destination, the UDM (dest=UDM, protocol=HTTP/2).

Table 3: Typical examples of question-answering tasks

Question Answering Task	Prompts	Answers
What-if Analysis	**Task's Prompt Template**: Please do what-if analysis: this task aims to forecast potential events across various aspects under hypothetical conditions, primarily defined by the presence or absence of certain events. In cases where new alert correlations emerge due to these conditions, it is necessary to explain these correlations. Conversely, if no new alert correlations are detected, the outcome would be a report of no new findings. **Question**: What if I solve the Kubescape breaches (privileged container and sensitive mount) under the current context? **Context**: [Falco]: Launch_Privileged_Container (user: test, container:attack-nginx, ns:free5gc, image:nginx), Launch_Sensitive_Mount (user: test, host_path:/var/run/docker.sock, verb: create) [Kubescape]: Privileged_Container (container:attack-nginx, ns:free5gc), Sensitive_mount (mount=/var/run/docker.sock) [5G Audit]: Steal_Data_from_UDM (an HTTP/2 request to UDM from unexpected IPs other than NFs' IPs) [5G Detect]: Suspicious HTTP/2 traffic from an unknown IP (dest: UDM, protocol: HTTP/2)	Kubescape's breaches involve a privileged container and a sensitive mount, which correspond to Falco's alerts of launching a privileged container and a sensitive mount. The correlation is evident as the privileged container reported in Kubescape's breach shares the same image (nginx) and username (test) as the one in Falco's alerts. Similarly, the sensitive mount has the same file path (/var/run/docker.socket) and username (test) in both Kubescape's breach and Falco's alerts. This indicates that the occurrences in Kubescape are directly linked to and caused by the scenarios detected in Falco's alerts. Furthermore, there is an implication that if these Kubescape and Falco alerts and breaches were to cease, related 5G alerts and breaches, such as data theft from HTTP and suspicious HTTP traffic, would also diminish.

Table 4: Typical examples of question-answering tasks (continued from the previous page)

Chapter 5

IMPLEMENTATION

This chapter presents 5GSecRec testbed implementation and the details of its components and integration.

5.1 Testbed Implementation

We implement a 5G testbed with several security tools as follows.

5.1.1 5G Core Network

To implement the 5G core network in a container environment, we utilize free5GC [102] to simulate the 5G core and deploy it on Kubernetes (K8S) [6] by Towards5GS-helm [103] within a Vagrant VM. The RAN and UE are simulated using UERANSIM [104] for executing 5G functionalities. Free5GC is installed and deployed atop Kubernetes within a Vagrant virtual machine, which is hosted on a remote OpenStack [105] server. UERANSIM is encapsulated in a Kubernetes pod, enabling it to communicate with 5G network function pods within the "free5gc" namespace.

5.1.2 Security Tools

For both K8S and 5G, we leverage existing open-source security tools as well as our implemented customized security solutions. For K8S, we use open-source tools, Falco [8] for detection on system calls and Kubernetes audit logs, and Kubescape [9] for verification on K8S configurations (e.g., role bindings, cronjobs, etc.). Implementation and deployment details of Falco and Kubescape are described in Section 7.2. As there currently exists no open-source auditing/detection tool for 5G, we implement an LSTM-based detection tool, which analyzes 5G traffic logs collected by Kubeshark [106] in the .pcap format, and a script-based auditing tool to verify custom rules on 5G network functions. Their details are shown as follows:

- *LSTM-based detection tool.* First, we deploy Kubeshark in the "free5gc" namespace, which exclusively contains pods functioning as 5G network functions. Kubeshark captures the network communications occurring within these 5G network functions, outputting the data as .pcap files. We then assign the names of the network functions to the IP field of each traffic packet, based on the IP addresses assigned to these network functions. Following this, we train an LSTM model using a dataset obtained from [107], which also forms the basis for the 5G-level attacks implemented in this work. The process involves implementing the 5G-level attacks, utilizing Kubeshark to collect traffic packets, and then employing the trained LSTM model to determine whether any traffic packets in the collected data are abnormal.

- *Custom rules for auditing 5G.* We collect application logs from the network function pods within free5gc using the command "kubectl logs", which includes all 5G-level events. The rules we have defined are tailored to identify the footprints of the 5G-level attacks as described by [107, 108]. For example, in the case of the "AMFLookingForUDM" attack, the logs will show an NRF Discovery event similar to "http://$NRF's IP$/nnrf-disc/v1/nf-instances?requester-nf-type=$randomString&target-nf-type=...". This attack takes advantage of the absence of input validation in free5gc. To verify if the 5G functions are compromised, we validate the values of "requester-nf-type" and "target-nf-type" in these requests.

5.1.3 Implementation Challenges

While implementing the 5G testbed, we encountered several challenges as follows. (i) While deploying security solutions for K8S, we faced a memory limit issue and hence reconfigured the *kube-apiserver.yaml* file and *set–audit-webhook-mode* to blocking so that the K8S API server does not exhaust the limited memory. (ii) When configuring the audit policy for K8S, the initial approach is to audit every security event captured by the K8S API, without filtering events based on resources, levels, groups, etc. This approach can lead to an overload of non-essential security events, which may overwhelm and cause failures in the webhook backend, subsequently disrupting Falco's monitoring process. By refining the audit policy (e.g., setting verbs to [watch, list, create, update, delete], etc.) to capture those security events that might trigger Falco alerts as required, Falco's monitoring can be maintained. (iii) Also, we have encountered a significant challenge in the deployment of the 5G core network: the UPF pod fails to start due to issues with its N6 interface. Specifically, an IP address cannot be bound to the N6 interface, a problem attributed to the network plugins. To resolve this issue, we found it necessary to manually configure the N6 interface to eth0. This manual intervention allows the UPF pod to be successfully created. This addresses the limitation in the network plugin's automatic configuration and ensures the proper functioning of the UPF pod.

5.2 5GSecRec Implementation

This section describes the 5GSecRec architecture and its implementation details.

5.2.1 System Architecture

Figure 10 shows a high-level system architecture of 5GSecRec with two major modules.

- *Model Training Module.* This module takes the security results from the auditing and monitoring system as inputs and feeds into the data processing component which encompasses a log processor that refines raw logs, succeeded by a log processor that concatenates multiple event logs sequentially. This concatenated data is subsequently employed to train a multi-modal model. Additionally, there is a question-answering Model Builder that exploits a fine-tuning data generator. This generator, based on experts' knowledge, produces question-answering pairs which are sent to a Model Trainer for fine-tuning LlaMA-2.

- *Inference and Response Module.* This module starts with the Prompt Builder module that takes in the user's queries and processes them using an NLP sentiment analyzer. This analyzer's role is crucial as it classifies the user's queries into distinct tasks, selecting appropriate templates for each. Following this classification, an alert correlator is involved to receive the security results to determine if any alerts or security breaches from the security data are correlated to the alerts or breaches selected by users. Once alert correlations are found, the Recommendation Generator is activated. It utilizes the capabilities of LLM text-generation pipelines powered by LangChain [109] to craft responses or recommendations.

5.2.2 Implementation Details

The Hugging Face [110] transformer library is utilized for the implementation of fine-tuning, multi-modal learning, and self-supervised learning. This library offers classes such as trainer, AutoTokenizer, and DataCollator, enabling the efficient downloading, training, saving, and using LLMs. For multi-modal correlation and question-answering tasks, the LlaMA-2 model [100] with both encoder and decoder architecture is selected as it is an open-source LLM and has a comparable performance with GPT-4 on certain NLP tasks. In our initial approach to learning correlations from historical data, we plan to utilize efficient models like BERT and RoBERTa. These models are selected for their ability to minimize computational resources while still delivering robust performance in NLP tasks. We recognize the trade-off between reducing computational costs and potential performance impacts. Additionally, it is noted that fine-tuning large language models (LLMs), particularly more resource-intensive ones like LlaMA-2, with specific knowledge such as alert correlations, is often inefficient. However, we discover a method mentioned in Section 5.2.3 to lower the high computational demands of fine-tuning these larger LLMs, allowing us to leverage their advanced capabilities. Moreover, we also consider other open-source LLMs as feasible alternatives. However, given the complexity and difficulty of our question-answering and correlation tasks and the demonstrated performance of LlaMA-2, we believe that LlaMA-2 is well-suited for testing our solution. The implementation of training LlaMA-2 by Hugging Face is shown as follows:

- *Load a pre-trained model and its tokenizer.* We first use the *AutoTokenizer.from_pretrained()* function to download and cache the tokenizer. It's important to set *tokenizer.pad_token* to the

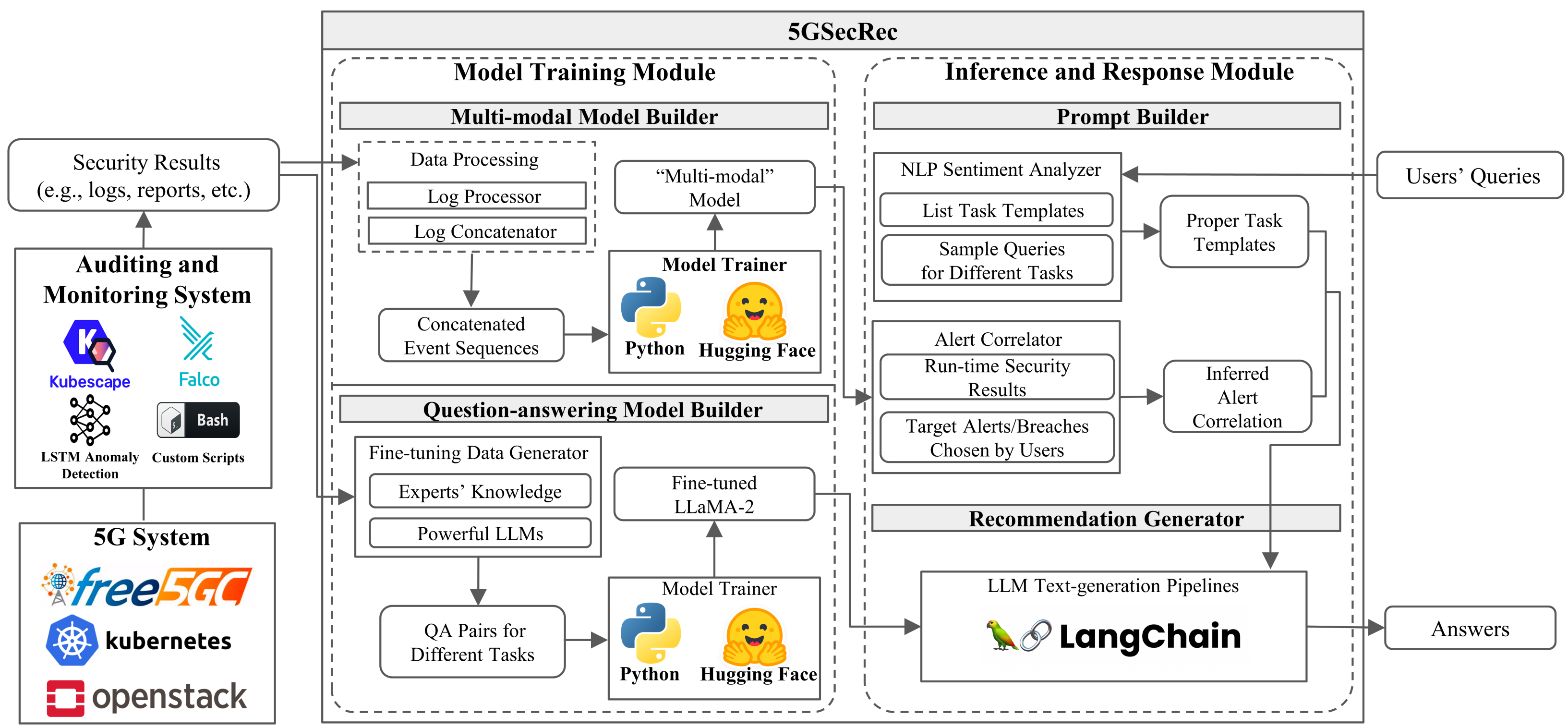


Figure 10: An overview of the high-level system architecture

eos_token, which represents the end-of-sequence token in LlaMA-2. Aligning the padding token with the model's understanding of sequence endings is crucial for correct input formatting. For the model itself, we utilize *AutoModelForCausalLM.from_pretrained()* to download and prepare LlaMA-2 for use.

- *Prepare the training and evaluation dataset.* Our training and evaluation datasets consist of input-output pairs. The *datasets.load_dataset()* function from the Hugging Face's Datasets library is used to load these datasets, ensuring they are in the correct format for model training and evaluation.

- *Define training arguments and initialize the training.* The *TrainingArguments()* function is used to define various training parameters, such as learning rate, batch size, gradient accumulation steps, and the number of epochs. We then instantiate the *Trainer* class with these arguments to kick off the training process. This class manages the training loop, handling the forward and backward passes, loss computation, model updates, and evaluation.

5.2.3 Implementation Challenges

During this implementation, we encountered the following challenges. The smallest LlaMA-2 model has around seven billion parameters, which require a lot of memory and computing power for both training and using the model. We often ran into memory problems during training, and the training process is slow when using all seven billion parameters across our GPUs. To solve this, we use parameter-efficient fine-tuning (PEFT) [111], specifically low-rank adaptation (LoRA) [112], which allows us to decrease the trainable parameters to 4.9 million from seven billion. This approach still provides good training performance, especially with a specific LoRA configuration (lora_alpha=16, lora_dropout=0.05, r=8). The number of training parameters decreases to 4,194,304 (trainable%: 0.062). Additionally, we employ Quantization [113] to reduce the precision of the numerical values to 4-bit in the model, which helps in using less memory and speeding up the process.

Chapter 6

EVALUATION

This chapter first describes our experimental setups and then presents the evaluation results of the effectiveness of 5GSecRec.

6.1 Experimental Setups

For our experiments, we use eight Quadro RTX 4000 GPUs with a total of 64GB VRAM to perform learning steps (i.e., training and evaluation). To prepare a training dataset, as shown in Table 5, we design attacks based on the MITRE ATT&CK framework [114], intended to trigger detection alerts in both Kubernetes and 5G verification tools. On Kubernetes, we deploy attack patterns from APT 29 [115] (T1087.02, T1110.03, T1005, T1068, T1083), APT17 [116] (T1070.004, T1057, T1083, T1059), and APT1 [117] (T1016, T1049, T1016, T1007, T1057, T1005). For 5G attacks [107, 108], we focus on Network Function (NF) level DoS attacks, compromising NF discovery services, executing Packet Forwarding Control Protocol (PFCP) deletions and modifications, and extracting subscriber data from UDMs. We use Caldera [118], a cybersecurity platform that focuses on automating adversary emulation, supporting manual red-team operations, and facilitating automated incident response. This tool is particularly notable for its use in simulating advanced persistent threat (APT) actors and helping organizations test and improve their defenses against such threats. We utilize it to automate the emulation of attacks on Kubernetes and 5G infrastructures.

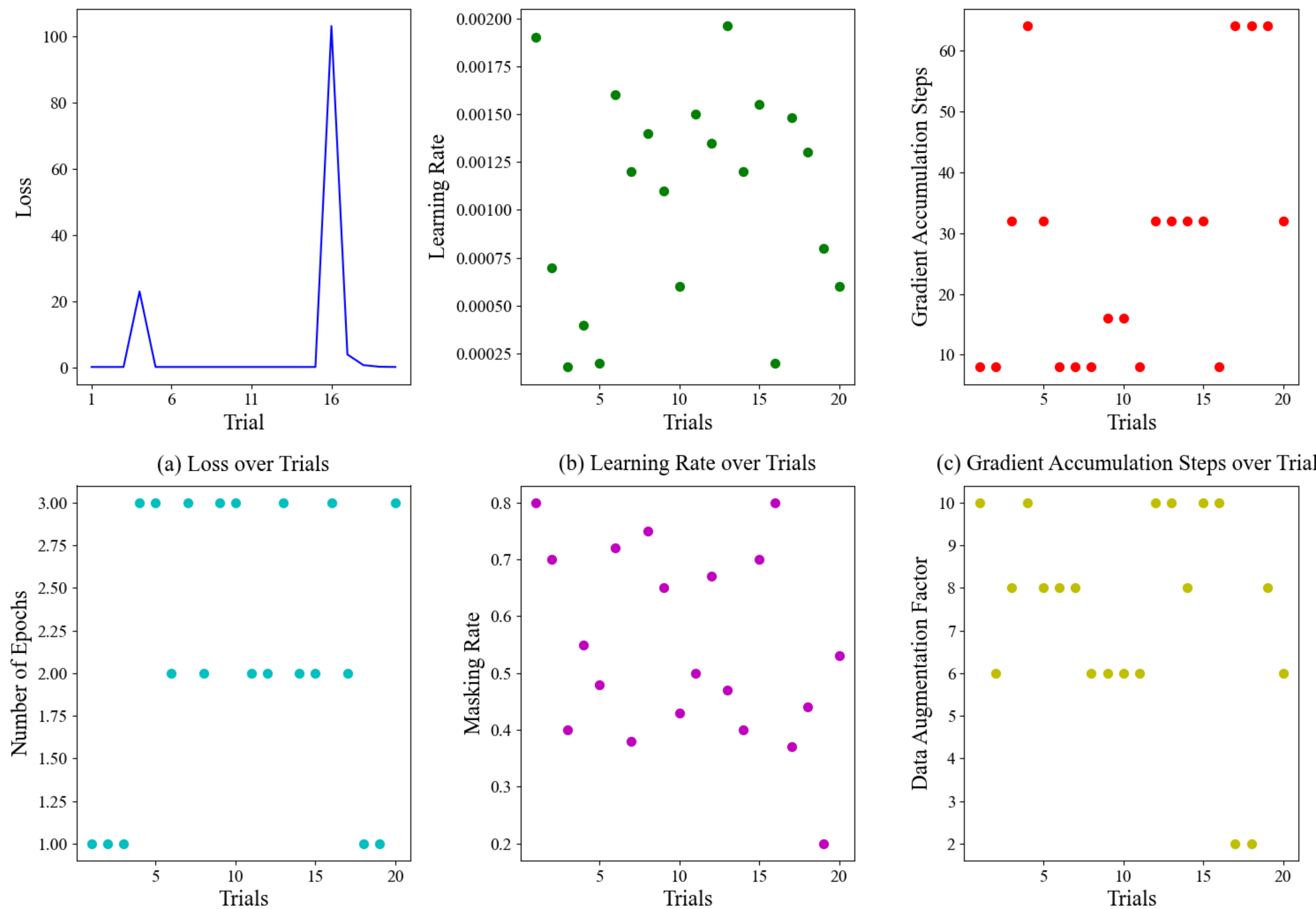
Since our 5G testbed implementation mentioned in Section 5.1.1 is based on a vagrant virtual machine, which is hosted on a remote OpenStack server, we first install Caldera on the OpenStack server to serve as a Command and Control (C2) server. This C2 server communicates with the vagrant machine, directing it to execute attack commands on the 5G testbed.

Attack Name	**Kubernetes Aspect**	**5G Aspect**
Custom Attack 1	APT 29 [115]: T1133, T1087.02, T1110.03, T1005, T1068, T1083	Network Function (NF) level DoS attacks [107]
Custom Attack 2	APT 17 [116]: T1070.004, T1057, T1083, T1059	Compromising NF discovery services (e.g., AMFLookingForUDM [107], etc.)
Custom Attack 3	APT 1 [117]: T1016, T1049, T1016, T1007, T1057, T1005	Compromising PFCP services [108], user data exfiltration from UDM [107]

Table 5: Summary of attacks on Kubernetes and 5G aspects

6.2 Hyperparameter Tuning

This set of experiments is to perform hyperparameter tuning for our solution. Figure 11 illustrates the Bayesian optimization output for identifying influential hyperparameters using the Tree-structured Parzen Estimator algorithm [119]. The examined hyperparameters encompassed gradient accumulation steps within $\{8, 16, 32, 64\}$; epochs within $\{1, 2, 3\}$, learning rates from 2×10^{-3} to 2×10^{-5}, masking rates from 0.2 to 0.8, and data augmentation factors within $\{2, 4, 6, 8, 10\}$. Specifically, Figure 11(a) correlates loss values with the optimal hyperparameter combination for up to 20 trials and finds 0.265 as the optimal loss value. Figure 11(b) indicates a broad exploration of learning rates, with 6.5×10^{-4} determined as optimal. Figure 11(c) contrasts the observed uniform value of gradient accumulation steps with the optimal value of 1. Figure 11(d) shows trails with a range for the number of epochs, where epoch=2 is the most optimal value. Figure 11(e) depicts a targeted range for the masking rate, with 0.44 identified as optimal. Figure 11(f) associates an augmentation factor of 2 with optimal model generalization, with no additional gains from higher values. These findings confirm the optimal hyperparameters' efficacy within specified ranges. The



(a) Loss over Trials (b) Learning Rate over Trials (c) Gradient Accumulation Steps over Trial

(d) Number of Training Epochs over Trials (e) Masking Rate over Trials (f) Data Augmentation Factor over Trials

Figure 11: Hyperparameter tuning results for the multi-modal model. (a) Loss over Trials (b) Learning Rate over Trials (c) Gradient Accumulation Steps over Trials (d) Number of Training Epochs (e) Masking Rate over Trials (f) Data Augmentation Factor over Trials

learning rate and masking rate exhibit precise optima, while epochs and gradient accumulation steps appear less sensitive. Data augmentation has a defined beneficial limit on model performance.

6.3 Evaluation on Alert Correlation

This set of experiments is to evaluate the accuracy of our alert correlation step using the multi-modal model. We employ three metrics to assess the performance of our multi-modal model. First, *perplexity* [120], is used to gauge the alignment between the model's predicted probability distribution and the actual word distribution in the text. Second, *accuracy* evaluates the model's accuracy in filling masked events given a partially incomplete event sequence. Finally, *F1 Score*, is used to provide a holistic assessment of the model's performance via the harmonic mean of precision and recall.

6.3.1 Evaluation Approach

After performing hyperparameter tuning, we choose the best combination of hyperparameters from Section 6.2 for training our multi-modal model. For evaluating the model's performance, we use about 10% of the dataset as the evaluation dataset and 90 % of the dataset as the training dataset. In this evaluation dataset, we applied within-aspect and across-aspect alert correlations to mask events within each ground-truth event sequence from the training data. This process resulted in creating two to three masked event sequences for each original event sequence. Note that all the masked event sequences in the evaluation dataset are ensured to be distinct from those in the training data. This evaluation dataset is designed to assess whether the model could correctly predict the missing events and complete the original event sequence. We utilize this dataset to test the performance of our multi-modal model by having it remove the masked events and generate the complete original event sequences. The summary of this dataset is shown in Table 6.

Size of Training Data	Size of Evaluation Data	Accuracy (%)	Perplexity	F1 Score
38,131	4,236	89.5	1.30	0.85

Table 6: Overall evaluation result of the multi-modal model

6.3.2 Evaluation Results

Figure 12 shows the performance of our multi-modal model in predicting the correlation among four different aspects (i.e., Kubernetes Monitoring (KM), Kubernetes auditing (KA), 5G Monitoring (5GM), and 5G Auditing (5GA)). Specifically, we evaluate the model's performance for ten prediction combinations ("Given KM, Predict 5GM", "Given KA, Predict 5GA", "Given KM, Predict KA", "Given KA, Predict KM", "Given 5GM, Predict KM", "Given 5GA, Predict KA", "Given 5GM, Predict 5GA", "Given 5GA, Predict 5GM", "Given 5GM & 5GA, Predict KM & KA", and "Given KM & KA, Predict 5GM & 5GA"). We selected these ten combinations based on the assumption that security analysts concentrate on both "within-layer" and "across-layer" event correlations. "Within-layer" correlation involves linking events from security tools operating on the same layer. For example, correlating events from Falco and Kubescape, both of which monitor the K8S layer, represents a "within-layer" correlation. On the other hand, "across-layer" correlation refers to connecting events from security tools deployed on different layers, such as correlating events between K8S and 5G layers. As shown in Figure 12, the tasks "Given KA, Predict 5GA" and "Given 5GM & 5GA, Predict KM & KA" demonstrate the highest accuracy and F1-scores among different scenarios, indicating that the models employed in these scenarios are highly effective.

Such high metrics suggest not only a high rate of correct predictions but also a strong balance between precision and recall. There is also a consistency in the performance across different predictive scenarios. For instance, the tasks "Given KM, Predict 5GM" and "Given 5GM, Predict KM" both display high accuracy (0.92) and robust F1 scores (0.84 and 0.87, respectively). This suggests that the models or methods used are versatile and robust, capable of adapting to various types of predictions.

However, the task "Given KM, Predict KA" presents a lower accuracy (0.78) and F1-score (0.74) compared to others. This indicates that this predictive scenario might be more challenging,

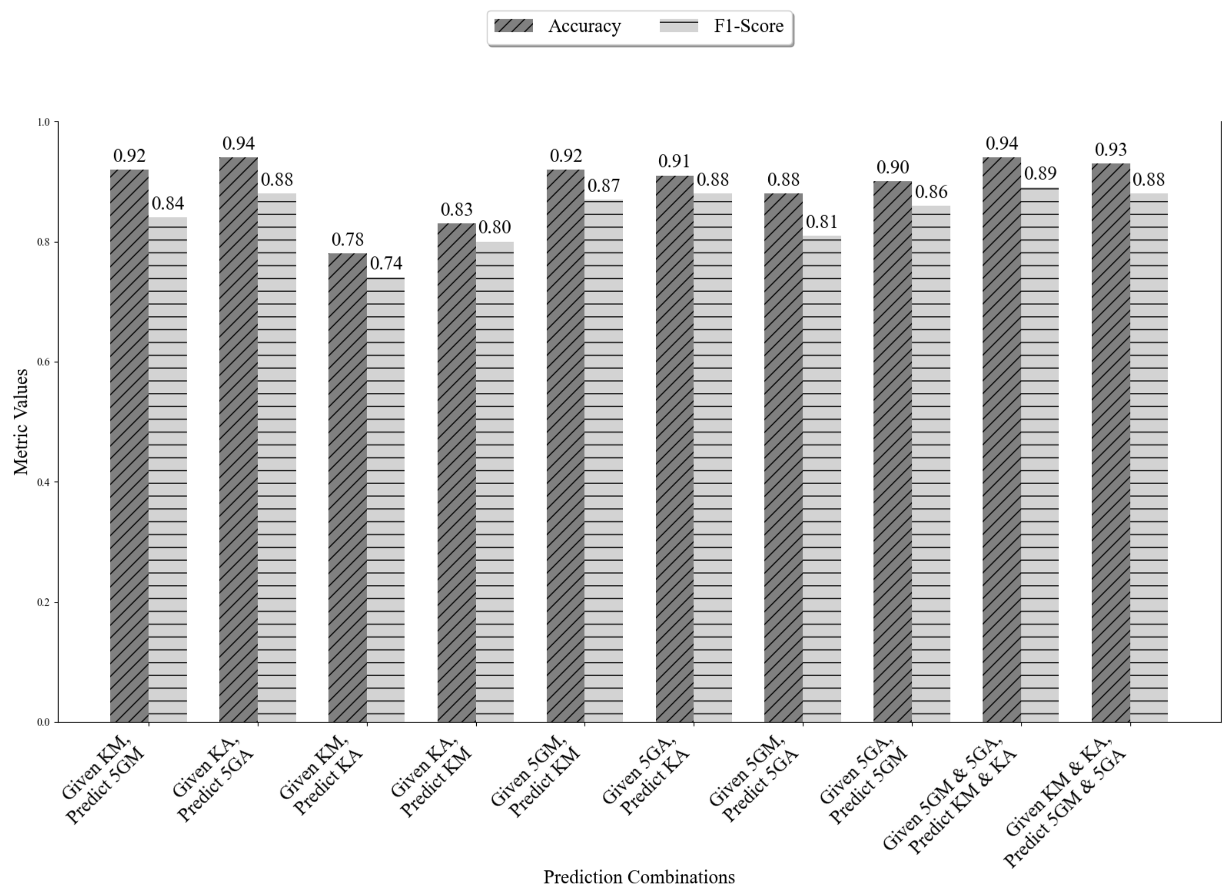


Figure 12: Performance of multi-modal model for predicting alert correlation (KM: Kubernetes Monitoring; KA: Kubernetes Auditing; 5GM: 5G Monitoring; 5GA: 5G Auditing)

or the models used are not as effective in this context, or KM and KA have more diverse event types and details than those of 5GM and 5GA, or the training dataset is limited. Most of the predictive tasks, including "Given KA, Predict KM" and "Given 5GA, Predict 5GM," showcase a good balance between accuracy and F1-score. This balance is crucial as it suggests that the models are not just accurate but also maintain a good balance between precision and recall, ensuring the reliability of the predictions.

6.4 Evaluation on Question-Answering

This set of experiments is to measure the quality of the question-answering step of our solution.

6.4.1 Evaluation Metrics.

To assess the performance of the LLM in question-answering, we utilize three distinct metrics. First, we use BLEU (Bilingual Evaluation Understudy) [121] to gauge the quality of text produced by machine translation systems. Second, METEOR (Metric for Evaluation of Translation with Explicit ORdering) [122] is employed to examine the linguistic alignment between the candidate and reference sentences, offering a correlation with human judgments. Finally, BERTscore [123] is adopted to determine the similarity between the candidate text and the reference text, leveraging the cosine similarity of pre-trained word embeddings in the sentences.

6.4.2 Experimental Approaches.

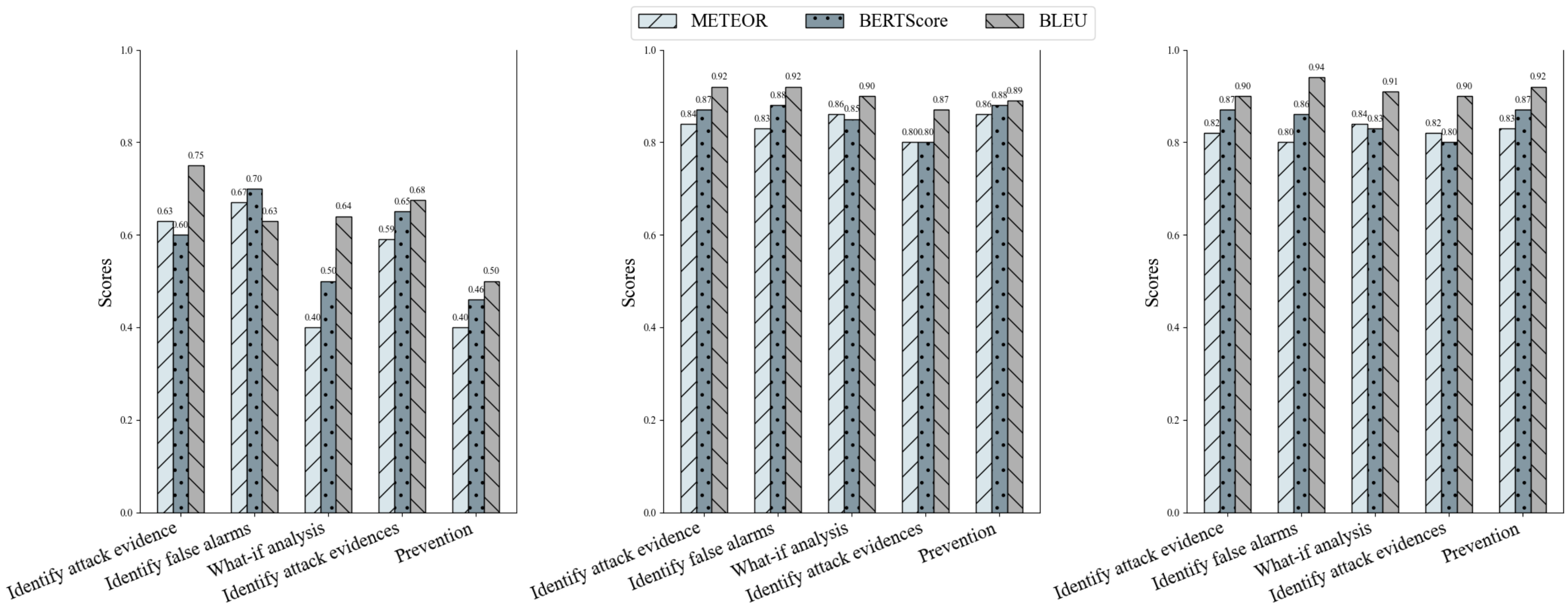
For this set of experiments, we designed three approaches. 1) We directly use fine-tuned LLM for question-answering with the input format as follows: *task template* + *"Context: run-time alerts/events"* + *"External References:"*. 2) We use fine-tuned LLM coupled with the multi-modal model's outputs with the input format: *task template* + *"Context: Multi-modal model's outputs"* + *"External References:"*.3) We use fine-tuned LLM coupled with multi-modal model outputs but with a different input format: *task template* + *"Context: run-time alerts/events"* + *"External References: Multi-modal model's outputs"*. Approach 1 serves as a baseline. Approach 2 aims to assess if the event correlations predicted by the multi-modal model enhance question-answering tasks. Approach 3 retains these predicted event correlations from the multi-modal model and introduces raw logs alongside the event correlations. This is to determine if the raw logs, which offer richer but less precise context compared to the model's event correlations, further improve question-answering tasks. ”External References” and ”Context” act as separators, ensuring that inputs are clear and comprehensible to the QA-LLM.

6.4.3 Evaluation Results.

Figure 13 (a) shows that the scores for all metrics fall between 0.4 and 0.75. The “Identify false alarms” task has the top METEOR Score at 0.75, while the lowest is seen in the “Identify Attack Evidence” task at 0.5. The BERTScore is quite steady across tasks, with scores ranging from 0.6 to 0.67. In contrast, the BLEU Score has more variation, with the highest being 0.7 for “Identify false alarms” and the lowest at 0.4 for both “What-if analysis” and “Prevention.” In Figure 13 (b), the scores are again between 0.4 and 0.75. The METEOR Score is highest for the “What-if analysis” task at 0.94. BERTScore remains stable across different tasks, showing scores from 0.82 to 0.87. The BLEU Score is fairly consistent, though a bit lower than BERTScore, with scores from 0.8 to 0.92.

Figure 13 (c) presents scores that are similar to Figure 2, mainly between 0.8 and 0.92. The top METEOR Score is 0.92, seen in both the “Identify false alarms” and “What-if analysis” tasks. BERTScore continues its stable performance with scores from 0.83 to 0.9. The BLEU Score varies from 0.8 to 0.89, with the “Prevention” task scoring the highest.

The results above indicate that merely fine-tuning the QA-LLM is insufficient for accurately recalling the event correlations necessary to answer user questions, leading to answers that may not meet expectations. This limitation could stem from the focus on fine-tuning data on question-answering tasks, rather than on correlating events. The fine-tuning process primarily trains the QA-LLM to respond to users' questions based on specific prompts. Hence, it is challenging for the QA-LLM to capture the underlying event correlations when using raw logs as context. Especially, for ”What-if analysis”, a hypothetical situation diminishes the performance of the QA-LLM. Using the outputs from the multi-modal model, both Approach 2 and Approach 3 improve the QA-LLM's performance. This is because they provide more precise event correlations, enabling the QA-LLM



(a) Approach 1: Fine-tuning Only

(b) Approach 2: Fine-tuning + Multi-modal model

(c) Approach 3: Fine-tuning + Multi-modal model
(with a variant prompt format)

Figure 13: Evaluation on question-answering with our different approaches

to respond directly with insights to users' questions based on these correlations, rather than having to infer event correlations from raw logs. However, there is no significant difference in the performance between Approach 2 and Approach 3. This could be due to the noise in the raw logs in contrast to the more refined outputs of the multi-modal model.

An alternative approach involves training the QA-LLM to understand both event correlations and question-answering tasks. This would require labeling event correlations and compiling them with question-answering patterns into question-answer pairs. These pairs would then be used to teach the QA-LLM to map relationships among events across the four aspects. However, this method faces two significant challenges: (i) the scarcity of labeled datasets, and (ii) overheads mentioned in Section 6.6. More specifically, within a question-answer pair, the event mappings are structured such that on the question side, there are events from one or multiple aspects, along with the questions. On the answer side, there are responses to these questions, which incorporate the correlated events. We systematically enumerate these mappings.

6.5 Experiments with ChatGPT

This set of experiments is to compare our solution with ChatGPT (GPT-3.5). Table 7 compares the question-answering capabilities of ChatGPT (GPT-3.5 without multi-modal model results) and our solution in an identify-attack-evidence task. In the first instance, ChatGPT fails to confirm a real attack, showing a lack of understanding of the context. In the second instance, it does not provide a meaningful analysis. However, our solution can identify the absence of a real attack, demonstrating its effectiveness. Differences observed in the responses of ChatGPT compared to our solution could likely be due to ChatGPT's lack of domain-specific knowledge such as alert correlations of the four aspects. ChatGPT depends on its general pre-trained knowledge and struggles to respond accurately to domain-specific questions. For example, it uses the basic concept that attacks often begin with privilege escalation to infer that a combination of a privileged container and a sensitive mount could lead to attacks in 5G such as data theft from UDM and suspicious HTTP traffic.

Table 8 presents a comparison between ChatGPT (GPT-3.5 with multi-modal model results) and our solution in responding to the same question in Table 7. Although ChatGPT's responses differ in style from what we expected based on the fine-tuning data, both ChatGPT and our solution display comparable accuracy, correctly identifying the absence of a real attack due to a missing Kubescape breach (privileged container), indicating their understanding of the context. However, a significant concern arises with data security: ChatGPT operates on third-party servers, which could pose risks to privacy and confidentiality.

Questions with the context	**Responses**	
	ChatGPT (GPT-3.5)	**Our solution**
Given Falco events and alerts from other tools, is there sufficient evidence to confirm a real attack? Context:[Falco] Launch_Priviledged_Container, Launch_Sensitive_Mount, other events [Kubescape] Sensitive_mount, other events [5G verification] Steal_Data_from_UDM, other events [5G detection] Suspicious_HTTP traffic. External Knowledge:	Yes, there is strong evidence to confirm a real attack. The correlation between Falco events (privileged container and sensitive mount), and 5G alerts (data theft from UDM and suspicious HTTP traffic) points to a coordinated attack involving unauthorized access, sensitive data theft, and potentially malicious network activity. This pattern of events strongly indicates a real attack.	No, Falco alerts (launch privileged container) should be related to Kubescape breaches (Privileged container), both of them together lead to 5G attacks like suspicious HTTP traffic. Hence, it is not a real attack.
	ChatGPT (GPT-3.5)	**Our solution**
Given Falco events and alerts from other tools, is there sufficient evidence to confirm a real attack? Context:[Falco] Launch_Priviledged_Container, Launch_Sensitive_Mount, other events [Kubescape] Sensitive_mount, other events [5G verification] Steal_Data_from_UDM, other events [5G detection] Suspicious_HTTP traffic. External Knowledge:	Based on your provided information, I can not analyze the context and answer the question	No, Falco alerts (launch privileged container) should be related to Kubescape breaches (Privileged container), both of them together lead to 5G attacks like suspicious HTTP traffic. Hence, it is not a real attack.

Table 7: Comparisons of question-answering between ChatGPT (GPT-3.5 without multi-modal model's results) and our solution

We can further refine ChatGPT to respond in the desired question-answering style, using a fine-tuning process similar to that used for LLaMA-2. Given LLaMA-2's comparable capabilities in

Questions with the context	Responses	
	ChatGPT (GPT-3.5)	**Our solution**
Given Falco events and alerts from other tools, is there sufficient evidence to confirm a real attack? Context: [Falco] Launch_Privileged_Container, Launch_Sensitive_Mount, other events [Kubescape] Sensitive_mount, other events [5G verification] Steal_Data_from_UDM, other events [5G detection] Suspicious_HTTP traffic. External Knowledge:	No, it is missing a Kubescape breach [Privileged container]	No, Falco alerts (launch privileged container) should be related to Kubescape breaches (Privileged container), both of them together lead to 5G attacks like suspicious HTTP traffic. Hence, it is not a real attack.
	ChatGPT (GPT-3.5)	**Our solution**
What if I delete Launch Privileged Container? Context:[Falco] Launch_Priviledged_Container, Launch_Sensitive_Mount, other events [Kubescape] Sensitive_mount, other events [5G verification] Steal_Data_from_UDM, other events [5G detection] Suspicious_HTTP traffic. External Knowledge:	Based on your description, you mentioned the event Launch Privileged Container and are considering what would happen if you were to delete it from the list of events. Here's how this change might impact the analysis: Presence of Launch Privileged Container: If Launch Privileged Container is included in the list of monitored events, the system will generate alerts when such an event occurs. You would then analyze the correlations between these alerts and other events to identify potential patterns or security threats. Absence of Launch Privileged Container: If you remove Launch Privileged Container from the list of monitored events, the system will no longer generate alerts for this specific event. Consequently, any correlations involving this event will no longer be detected, and the analysis will focus on the remaining events.	If Launch Privileged Container is deleted, Falco alerts, such as Launch Sensitive Mount, and Kubescape breaches, such as 'Sensitive Mount,' will still be generated. However, the 5G audit results, such as 'Steal Data from UDM,' and the 5G detection results, such as 'Suspicious HTTP traffic,' will no longer be generated

Table 8: Comparisons of question-answering between ChatGPT (GPT-3.5 with multi-modal model's results) and our solution

natural language processing tasks [100], we anticipate that ChatGPT will meet our expectations in responding to user inquiries. However, concerns about data confidentiality and privacy when transmitting data to ChatGPT remain unresolved. Therefore, we do not perform comparisons between the outputs of the fine-tuned ChatGPT and our solution.

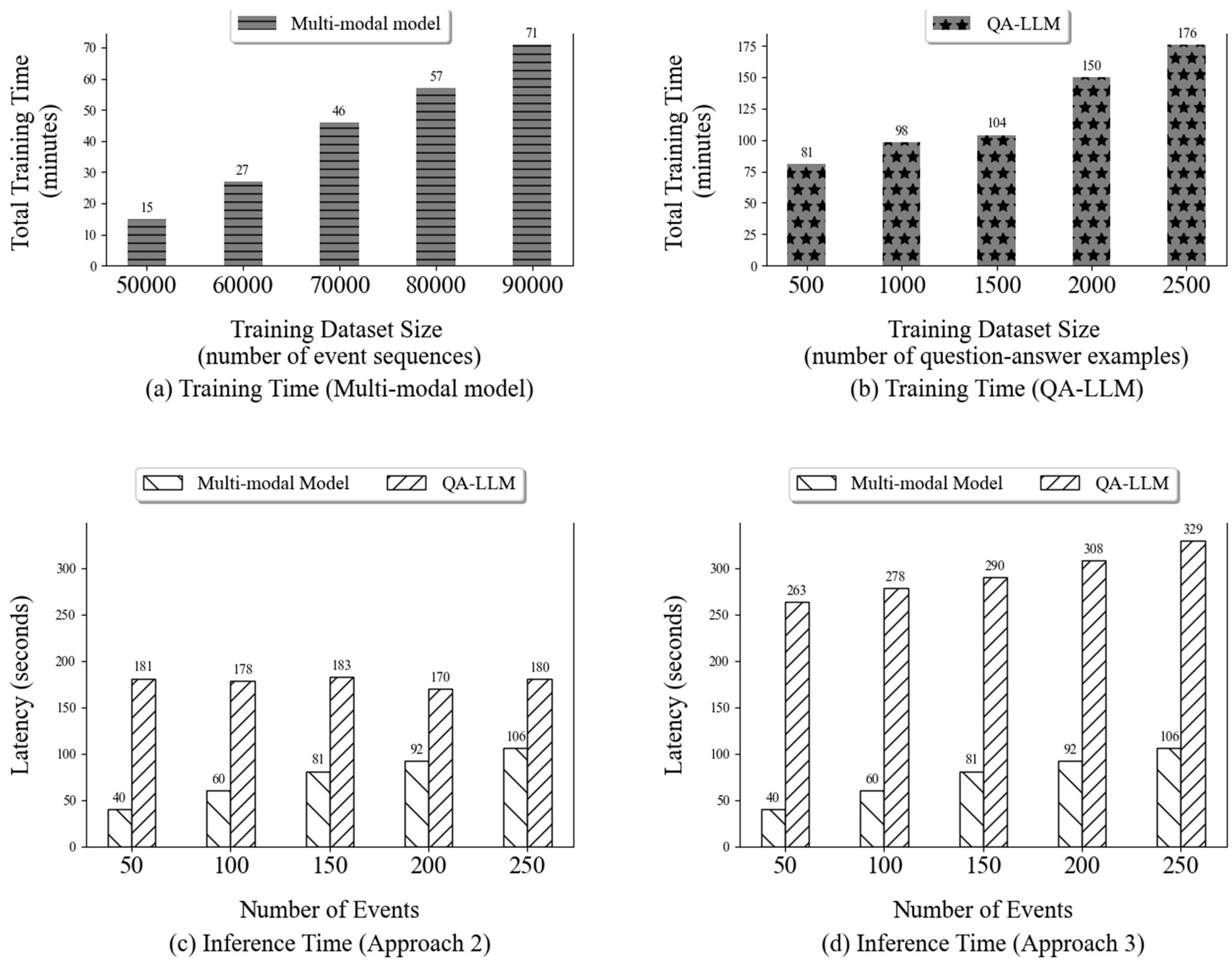
6.6 Overheads

The final set of experiments is to measure the overhead of our solution.

6.6.1 Training Time

Figures 14(a) and 14(b) show the training times of our multi-modal model and QA-LLM. For the multi-modal model, training times increase as the dataset size grows, starting at 15 minutes for a dataset size of $50,000$ and going up to 71 minutes for a dataset size of $90,000$. These events are gathered using our 5G testbed, which utilizes free5GC. After pre-processing, each event typically consists of about 50 tokens on average. In a dataset of 50,000 events, we accumulate approximately 1000 time windows, each containing an average of 50 events across the four aspects. This suggests a proportional increase in training time with larger datasets. In contrast, the QA-LLM requires more time to train across all dataset sizes, with training times beginning at 81 minutes for the smallest dataset and reaching 176 minutes for the largest. The higher training times for the QA-LLM compared to the multi-modal model could be attributed to factors such as the number of tokens in the training data.

However, it is not optimal to enumerate mappings of events from different aspects in question-answer pairs and use them to train a QA-LLM. Although this approach can teach the QA-LLM to learn labeled event correlations across different aspects and question-answering simultaneously, the scalability is limited. As the number of aspects grows, the increase in enumeration substantially enlarges the size of the question-answer pairs and extends the training time.



(a) Training Time (Multi-modal model)
(b) Training Time (QA-LLM)
(c) Inference Time (Approach 2)
(d) Inference Time (Approach 3)

Figure 14: Training time and running time for the multi-modal model and QA-LLM

6.6.2 Inference Time

Figures 14(c) and 14(d) compare the inference times of these two models for Approaches 2 and 3, respectively. In Approach 2, both models process only the outputs of the multi-modal model. Here, the QA-LLM consistently shows higher latency than the multi-modal model, indicating a slower processing capability. The latency for the multi-modal model ranges from 40 to 106 seconds, whereas for the QA-LLM, it ranges from 170 to 183 seconds across various numbers of events. Approach 3 introduces a variation where runtime events from the multi-modal model's outputs are added to the prompt for the QA-LLM. This change results in increased prompt lengths and, consequently, longer inference times for the QA-LLM. As the throughput increases, the QA-LLM's inference time also increases, starting at 263 seconds and going up to 329 seconds. This rising trend in latency with increasing throughput is not observed in the multi-modal model, suggesting that the QA-LLM's processing is more significantly affected by the increased prompt length in Approach 3. Since the difference in their effectiveness for answering questions is minimal, Approach 2 is the more suitable option. Although the QA-LLM takes minutes to generate answers based on event correlation, this is more efficient compared to the hours a security analyst would spend correlating events and deriving insights from them.

Chapter 7

OTHER CONTRIBUTIONS

This Chapter focuses on the other contributions made to this thesis.

7.1 Overview of Our Work on Security Posture Evaluation for 5G Networks

Assessing 5G network security poses significant challenges due to its scale and new security threats. Traditional methods, often yielding simple binary outcomes, fail to capture the network's complex security status. This work introduces an *event-state* model for evaluating 5G security, combining audit-derived **state models** with surveillance-based **event models**. The state model identifies attacker-induced non-compliances, while the event model monitors attacker actions. These models are fused *horizontally* to correlate audit and surveillance findings, and *vertically* across different network layers. The culmination is a probabilistic model using Bayesian networks, offering a nuanced view of 5G network security. Thus, this work proposes a framework for security posture evaluation in 5G networks (namely, 5GSPE [124]).

7.2 Implementation of 5GSPE

The main contribution to this thesis from 5GSPE is based on the implementation and deployment of security tools (Falco, and Kubescape).

- *Falco [8].* Falco is a cloud-native security solution, open-source and designed for continuous detection of threats and risks within Kubernetes, containers, and cloud-native environments. Its primary focus is on runtime security, which involves monitoring the activities of running applications and containers to maintain compliance with set rules. Initially, monitoring was implemented manually using *journalctl*, combined with a Kubernetes plugin for auditing Kubernetes events. Subsequently, the system transitioned to utilizing *docker run* in the least privileged permission configuration. This was facilitated by using the *falcosecurity/falco-driver-loader* image along with an eBPF probe. eBPF (Extended Berkeley Packet Filter) [125] represents a significant innovation in Linux kernel development, enabling the execution of sandboxed programs directly in the kernel, which was previously a concern for system stability and security. The adoption of an eBPF probe has made Kubernetes monitoring more effective, flexible, and secure. For alerting, the system configuration outputs alerts via syslog in JSON format.

- *Kubescape [9].* KubeScape is an auditing tool tailored for Kubernetes environments. Its core function is to analyze Kubernetes configurations and resource definitions, aligning them with established best practices and compliance standards. To perform an audit, KubeScape is initiated using the command "*Kubescape scan mitre*". This command triggers a verification process under the MITRE attack framework. Following the scan, any identified breaches are extracted using bash scripts. The results of this extraction are then compiled into a report, which is formatted in JSON and saved into a file.

Chapter 8

CONCLUSION

8.1 Summary

In cloud-native environments, the deployment of 5G services is structured across multiple layers. These layers range from the service layer, where network functions are instantiated in containers, to the virtualization layer, which manages infrastructure virtualization and resource provisioning. Although each layer has its specific function, they're interconnected for the sake of overall service delivery. Each layer may have security solutions tailored to its needs. However, due to differences in their auditing and monitoring methods, correlating outcomes from these diverse solutions becomes a challenge. The rapidity of incidents, often driven by automation, misconfigurations, and vulnerabilities, burdens SOC teams with an excessive number of alerts, among which a significant portion can be false positives. The traditional approach, relying on SOC teams' expertise to extract context from these correlations, frequently falls short.

In this thesis, we addressed these challenges by introducing 5GSecRec, an innovative solution that combines multi-modal learning, self-supervised learning, and Large Language Models (LLMs). 5GSecRec excels at identifying correlations in security alerts from both the virtualization and service layers, providing actionable insights for security analysts. This solution was built on the foundations of free5gc and Kubernetes. We also implemented and deployed security tools targeting four key aspects of a cloud-native 5G system: Kubernetes monitoring, Kubernetes auditing, 5G monitoring, and 5G auditing. Our evaluation results demonstrated that our solution can effectively leverage LLMs to correlate alerts across these four aspects and provide domain-specific answers to questions about such event correlations.

8.2 Limitations and Future Works

One limitation of our study is the assumption that attackers cannot compromise the integrity of results from deployed security solutions. Should they manage to manipulate these results, our multi-modal model may struggle to accurately correlate alerts. Also, zero-day attacks often lead to unexpected combinations of alerts or breaches across different aspects. However, our work specifically excludes such scenarios if they present combinations that are distinct from those documented in historical results as training data. Furthermore, we currently focus on short-term alert correlations, without considering potential long-term dependencies between alerts from different aspects. In future research, we aim to address this by creating new self-supervised tasks to discern these long-term dependencies. Also, given our work's generic nature and its good performance with textual data correlation, we plan to expand its scope to include results of security solutions that cover layers (e.g., physical) other than *service layer* and *virtualization layer*. We will extend our work by further correlating 3GPP specifications as an extra reference with alerts. This aims to improve the interpretability of alert correlations and the reliability of the generated recommendation.

Bibliography

[1] CNCF, "Cncf cloud native definition v1.0." GitHub, 2018. Retrieved 2023-10-10.

[2] P. Jiang, Q. Wang, M. Huang, C. Wang, Q. Li, C. Shen, and K. Ren, "Building in-the-cloud network functions: Security and privacy challenges," *Proceedings of the IEEE*, vol. 109, no. 12, pp. 1888–1919, 2021.

[3] VMware, "VMware expands its VMware ready for telco cloud program to accelerate the deployment of 5G services," 2020. Available at:t.ly/BIIW.

[4] E. S. for NFV, "Network functions virtualisation." https://www.etsi.org/technologies/nfv/, 2023. Accessed on: Oct 10, 2023.

[5] Docker, "Docker documentations." https://docs.docker.com//, 2023. Accessed on: Oct 10, 2023.

[6] Kubernetes, "Kubernetes." https://kubernetes.io/, 2023. (Accessed: 10 Oct 2023).

[7] Ericsson, "5G Core (5GC) network: Get to the core of 5G." https://www.ericsson.com/en/core-network/5g-core, Dec. 2022. (Accessed: 24 May 2023).

[8] Falco, "Falco." https://github.com/falcosecurity/falco, Dec. 2023.

[9] Kubescape, "Kubescape." https://github.com/kubescape/kubescape, Dec. 2022.

[10] H. T. Elshoush and I. M. Osman, "Intrusion alert correlation framework: An innovative approach," in *IAENG Transactions on Engineering Technologies: Special Volume of the World Congress on Engineering 2012*, pp. 405–420, Springer, 2013.

[11] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on dependable and secure computing*, vol. 1, no. 3, pp. 146–169, 2004.

[12] E. Totel, B. Vivinis, and L. Mé, "A language-driven intrusion detection system for event and alert correlation," in *Security and Protection in Information Processing Systems: IFIP 18 th World Computer Congress TC11 19 th International Information Security Conference 22–27 August 2004 Toulouse, France*, pp. 209–224, Springer, 2004.

[13] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?," *arXiv preprint arXiv:1909.01066*, 2019.

[14] F. Döringer Kana, "Synthetic data generation for domain adaptation of a retriever-reader question answering system for the telecom domain: Comparing dense embeddings with bm25 for open domain question answering," 2023.

[15] "Adopting neural language models for the telecom domain," January 2022. Accessed: 2023-08.

[16] OpenAI, "Chatgpt." `https://openai.com/chatgpt`, Aug 2023. Accessed: 2023-08.

[17] X. Wu, R. Duan, and J. Ni, "Unveiling security, privacy, and ethical concerns of chatgpt," *Journal of Information and Intelligence*, 2023.

[18] G. Sebastian, "Privacy and data protection in chatgpt and other ai chatbots: Strategies for securing user information," *Available at SSRN 4454761*, 2023.

[19] National Vulnerability Database (NVD), "Cve-2023-3893 detail." `https://nvd.nist.gov/vuln/detail/CVE-2023-3893`, 11 2023. A security issue was discovered in

Kubernetes where a user that can create pods on Windows nodes running kubernetes-csi-proxy may be able to escalate to admin privileges on those nodes. Kubernetes clusters are only affected if they include Windows nodes running kubernetes-csi-proxy.

[20] ericsson, "Cloud native is transforming the telecom industry," 2023. Accessed: 2023-12-07.

[21] M. Yang, Y. Li, L. Hu, B. Li, D. Jin, S. Chen, and Z. Yan, "Cross-layer software-defined 5g network," *Mobile Networks and Applications*, vol. 20, pp. 400–409, 2015.

[22] ericsson, "Your guide to building a cloud-native infrastructure for 5g," 2023. Accessed: 2023-12-07.

[23] intel, "Cloud native 5g core," 2023. Accessed: 2023-12-07.

[24] Spirent, "Keeping Pace with the Requirements of 5G Security." https://www.spirent.com/assets/white-paper-keeping-pace-with-the-requirements-of-5g-security, July 2022. Publisher: Spirent.

[25] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan, "Sok: History is a vast early warning system: Auditing the provenance of system intrusions," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2620–2638, IEEE, 2023.

[26] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic literature review of security event correlation methods," *IEEE Access*, vol. 10, pp. 43387–43420, 2022.

[27] L. Wang, A. Liu, and S. Jajodia, "An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts," in *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*, pp. 247–266, Springer, 2005.

[28] L. Wang, A. Liu, and S. Jajodia, "Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts," *Computer communications*, vol. 29, no. 15, pp. 2917–2933, 2006.

[29] M. Yu and X. Zhang, "Alertinsight: Mining multiple correlation for alert reduction.," *Computer Systems Science & Engineering*, vol. 46, no. 2, 2023.

[30] J. D. Pereira, J. R. Campos, and M. Vieira, "An exploratory study on machine learning to combine security vulnerability alerts from static analysis tools," in *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, pp. 1–10, IEEE, 2019.

[31] J. D. Pereira, J. R. Campos, and M. Vieira, "Machine learning to combine static analysis alerts with software metrics to detect security vulnerabilities: An empirical study," in *2021 17th European Dependable Computing Conference (EDCC)*, pp. 1–8, IEEE, 2021.

[32] F. Mateo Tudela, J.-R. Bermejo Higuera, J. Bermejo Higuera, J.-A. Sicilia Montalvo, and M. I. Argyros, "On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications," *Applied Sciences*, vol. 10, no. 24, p. 9119, 2020.

[33] Y. Wang, Y. Guo, and C. Fang, "An end-to-end method for advanced persistent threats reconstruction in large-scale networks based on alert and log correlation," *Journal of Information Security and Applications*, vol. 71, p. 103373, 2022.

[34] H. Lyu, J. Liu, Y. Lai, B. Mao, and X. Huang, "Agcm: A multi-stage attack correlation and scenario reconstruction method based on graph aggregation," *Available at SSRN 4616395*.

[35] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, IEEE, 2019.

[36] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1172–1189, IEEE, 2020.

[37] S. Thirumuruganathan, M. Nabeel, E. Choo, I. Khalil, and T. Yu, "Siraj: a unified framework for aggregation of malicious entity detectors," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 507–521, IEEE, 2022.

[38] logrhythm, "Endpoint monitoring and threat detection." https://logrhythm.com/solutions/security/endpoint-threat-detection/, 2023. Accessed on: Oct 10, 2023.

[39] "McAfee Enterprise Security Manager." https://infosec.run/images/mcafee_enterprise_security_manager_11.2.x_product_guide_10-29-2019.pdf. (accessed: 14 May 2023).

[40] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[41] G. Shen, J. Jia, L. Nie, F. Feng, C. Zhang, T. Hu, T.-S. Chua, W. Zhu, *et al.*, "Depression detection via harvesting social media: A multimodal dictionary learning solution.," in *IJCAI*, pp. 3838–3844, 2017.

[42] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas, "Exploring hate speech detection in multimodal publications," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1470–1478, 2020.

[43] Y. Chen, D. Li, P. Zhang, J. Sui, Q. Lv, L. Tun, and L. Shang, "Cross-modal ambiguity learning for multimodal fake news detection," in *Proceedings of the ACM Web Conference 2022*, pp. 2897–2905, 2022.

[44] J. Venugopalan, L. Tong, H. R. Hassanzadeh, and M. D. Wang, "Multimodal deep learning models for early detection of alzheimer's disease stage," *Scientific reports*, vol. 11, no. 1, p. 3254, 2021.

[45] A. Rana and S. Jha, "Emotion based hate speech detection using multimodal learning," *arXiv preprint arXiv:2202.06218*, 2022.

[46] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[47] K. Bayoudh, R. Knani, F. Hamdaoui, and A. Mtibaa, "A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets," *The Visual Computer*, pp. 1–32, 2021.

[48] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2018.

[49] H. Moraliyage, V. Sumanasena, D. De Silva, R. Nawaratne, L. Sun, and D. Alahakoon, "Multimodal classification of onion services for proactive cyber threat intelligence using explainable deep learning," *IEEE Access*, vol. 10, pp. 56044–56056, 2022.

[50] A. Sedik, O. S. Faragallah, H. S. El-sayed, G. M. El-Banby, F. E. A. El-Samie, A. A. Khalaf, and W. El-Shafai, "An efficient cybersecurity framework for facial video forensics detection based on multimodal deep learning," *Neural Computing and Applications*, pp. 1–18, 2022.

[51] H. Zhang, G. Shen, C. Guo, Y. Cui, and C. Jiang, "Ex-action: Automatically extracting threat actions from cyber threat intelligence report based on multimodal learning," *Security and Communication Networks*, vol. 2021, pp. 1–12, 2021.

[52] X. Zhou and R. M. Verma, "Vulnerability detection via multimodal learning: datasets and analysis," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pp. 1225–1227, 2022.

[53] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren, "A novel multimodal-sequential approach based on multi-view features for network intrusion detection," *IEEE Access*, vol. 7, pp. 183207–183221, 2019.

[54] E. Garza, E. Hemberg, S. Moskal, and U.-M. O'Reilly, "Assessing large language model's knowledge of threat behavior in mitre att&ck," 2023.

[55] W. Tann, Y. Liu, J. H. Sim, C. M. Seah, and E.-C. Chang, "Using large language models for cybersecurity capture-the-flag challenges and certification questions," *arXiv preprint arXiv:2308.10443*, 2023.

[56] P. Mudgal and R. Wouhaybi, "An assessment of chatgpt on log data," 2023.

[57] H. Zhu, P. Tiwari, A. Ghoneim, and M. S. Hossain, "A collaborative ai-enabled pretrained language model for aiot domain question answering," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3387–3396, 2021.

[58] M. A. Arefeen, B. Debnath, and S. Chakradhar, "Leancontext: Cost-efficient domain-specific question answering using llms," *arXiv preprint arXiv:2309.00841*, 2023.

[59] Y. Wang, X. Ma, and W. Chen, "Augmenting black-box llms with medical textbooks for clinical question answering," *arXiv preprint arXiv:2309.02233*, 2023.

[60] V. Lomshakov, S. Kovalchuk, M. Omelchenko, S. Nikolenko, and A. Aliev, "Fine-tuning large language models for answering programming questions with code snippets," in *International Conference on Computational Science*, pp. 171–179, Springer, 2023.

[61] L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, and M. Debbah, "Understanding telecom language through large language models," *arXiv preprint arXiv:2306.07933*, 2023.

[62] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[63] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[64] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[65] H. Holm, "Bidirectional encoder representations from transformers (bert) for question answering in the telecom domain.: Adapting a bert-like language model to the telecom domain using the electra pre-training approach," 2021.

[66] W. Tong, C. Peng, T. Yang, F. Wang, J. Deng, R. Li, L. Yang, H. Zhang, D. Wang, M. Ai, *et al.*, "Ten issues of netgpt," *arXiv preprint arXiv:2311.13106*, 2023.

[67] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, and M. Debbah, "Large language models for telecom: The next big thing?," *arXiv preprint arXiv:2306.10249*, 2023.

[68] A. Maatouk, N. Piovesan, F. Ayed, A. De Domenico, and M. Debbah, "Large language models for telecom: Forthcoming impact on the industry," *arXiv preprint arXiv:2308.06013*, 2023.

[69] L. Ferreira, D. C. Silva, and M. U. Itzazelaia, "Recommender systems in cybersecurity," *Knowledge and Information Systems*, pp. 1–37, 2023.

[70] A. Pawlicka, M. Pawlicki, R. Kozik, and R. S. Choraś, "A systematic review of recommender systems and their applications in cybersecurity," *Sensors*, vol. 21, no. 15, p. 5248, 2021.

[71] N. Polatidis, E. Pimenidis, M. Pavlidis, and H. Mouratidis, "Recommender systems meeting security: From product recommendation to cyber-attack prediction," in *Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings*, pp. 508–519, Springer, 2017.

[72] N. Polatidis, E. Pimenidis, M. Pavlidis, S. Papastergiou, and H. Mouratidis, "From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks," *Evolving Systems*, vol. 11, pp. 479–490, 2020.

[73] F. Soldo, A. Le, and A. Markopoulou, "Predictive blacklisting as an implicit recommendation system," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, IEEE, 2010.

[74] S. McDonnell, O. Nada, M. R. Abid, and E. Amjadian, "Cyberbert: a deep dynamic-state session-based recommender system for cyber threat recognition," in *2021 IEEE aerospace conference (50100)*, pp. 1–12, IEEE, 2021.

[75] C. Sayan, S. Hariri, and G. Ball, "Cyber security assistant: Design overview," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pp. 313–317, IEEE, 2017.

[76] K. B. Lyons, "A recommender system in the cyber defense domain," 2014.

[77] M. F. Franco, B. Rodrigues, and B. Stiller, "Mentor: the design and evaluation of a protection services recommender system," in *2019 15th international conference on network and service management (CNSM)*, pp. 1–7, IEEE, 2019.

[78] E. Sula, M. Franco, and B. Rodriguez, "Protecddos: A recommender system for distributed denial-of-service protection services," *Bachelorthesis, CSG@ ifi, University of Zurich, Switzerland, Supervisors: Muriel Franco, Bruno Rodrigues*, 2019.

[79] J. Chen, L. Liu, R. Chen, W. Peng, and X. Huang, "Secrec: a privacy-preserving method for the context-aware recommendation system," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3168–3182, 2021.

[80] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE transactions on information forensics and security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[81] A. d. M. Del Esposte, R. Campiolo, F. Kon, and D. Batista, "A collaboration model to recommend network security alerts based on the mixed hybrid approach,"

[82] M. A. Ferrag, A. Battah, N. Tihanyi, M. Debbah, T. Lestable, and L. C. Cordeiro, "Securefalcon: The next cyber reasoning system for cyber security," *arXiv preprint arXiv:2307.06616*, 2023.

[83] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, and T. Lestable, "Revolutionizing cyber threat detection with large language models," *arXiv preprint arXiv:2306.14263*, 2023.

[84] 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.

[85] Y. X. ZXhang, Y. M. Haxo, and Y. X. Mat, "Falcon llm: A new frontier in natural language processing," *AC Investment Research Journal*, vol. 220, no. 44, 2023.

[86] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[87] P. Wang, Q. Wu, C. Shen, A. Dick, and A. Van Den Hengel, "Fvqa: Fact-based visual question answering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2413–2427, 2017.

[88] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7077–7087, 2021.

[89] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[90] Z. Huang, C. Lv, Y. Xing, and J. Wu, "Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11781–11790, 2020.

[91] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li, "Deep learning-based image segmentation on multimodal medical imaging," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 3, no. 2, pp. 162–169, 2019.

[92] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li, "Medical image segmentation based on multimodal convolutional neural network: Study on image fusion schemes," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 903–907, IEEE, 2018.

[93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[94] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[95] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[96] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Ground-truth adversarial examples," *arXiv preprint arXiv:1709.10207*, vol. 1, no. 1, pp. 2–2, 2017.

[97] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, *et al.*, "Technical report on the cleverhans v2. 1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2016.

[98] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[99] "Llm prompting guides." https://huggingface.co/docs/transformers/main/tasks/prompting, June 2023. Accessed: 2023-06.

[100] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, and et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023. Subjects: Computation and Language (cs.CL); Artificial Intelligence (cs.AI).

[101] L. Siddiqui, "The security analyst role: Skills, responsibilities & salary in 2023," 2023. Accessed: 2023-09-13.

[102] free5Gc, "free5GC." https://www.free5gc.org/, 2023. (Accessed: 07 Oct 2023).

[103] A. Khichane, I. Fajjari, N. Aitsaadi, and M. Gueroui, "Cloud native 5g: an efficient orchestration of cloud native 5g system," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2022.

[104] "Ueransim: Open source 5g ue and ran (gnodeb) simulator." https://github.com/aligungr/UERANSIM, 2022. Release 3.2.6. Retrieved January 13, 2022.

[105] OpenStack, "Open Source Cloud Computing Infrastructure - OpenStack." https://www.openstack.org/, 2023. (Accessed: 07 Oct 2023).

[106] "Kubeshark: The api traffic analyzer for kubernetes." https://github.com/kubeshark/kubeshark, 2023. Release 40.1, Retrieved May 4, 2023.

[107] C. Coldwell, D. Conger, E. Goodell, B. Jacobson, B. Petersen, D. Spencer, M. Anderson, and M. Sgambati, "Machine learning 5g attack detection in programmable logic," in *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 1365–1370, 2022.

[108] G. Amponis, P. Radoglou-Grammatikis, T. Lagkas, W. Mallouli, A. Cavalli, D. Klonidis, E. Markakis, and P. Sarigiannidis, "Threatening the 5g core via pfcp dos attacks: the case of blocking uav communications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, pp. 1–27, 2022.

[109] "Langchain: Building applications with llms through composability." https://github.com/langchain-ai/langchain, 2023. Version 0.0.227, Retrieved July 7, 2023.

[110] HuggingFace, "HuggingFace." https://github.com/huggingface/, May 2023.

[111] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods." https://github.com/huggingface/peft, 2022.

[112] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[113] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[114] Mitre, "MITRE ATT&CK." https://attack.mitre.org/, 2023.

[115] M. ATTCK, "APT29." https://attack.mitre.org/groups/G0016/, Oct. 2023.

[116] M. ATTCK, "APT17." https://attack.mitre.org/groups/G0025/, Oct. 2020.

[117] M. ATTCK, "APT1." https://attack.mitre.org/groups/G0006/, Oct. 2021.

[118] CALDERA, “CALDERA.” https://github.com/mitre/caldera, June 2023. Release date: June 19, 2023.

[119] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Advances in neural information processing systems*, vol. 24, 2011.

[120] S. F. Chen, D. Beeferman, and R. Rosenfeld, “Evaluation Metrics For Language Models,” 1 2008.

[121] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[122] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

[123] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” in *International Conference on Learning Representations*, 2020.

[124] M. N. Hoq, J. W. Yao, S. Majumdar, L. Suárez, L. Wang, A. Boukhtouta, M. Pourzandi, and M. Debbabi, “Evaluating the security posture of 5g networks by combining state auditing and event monitoring,” *Esorics*, 2023.

[125] “What is ebpf? - ebpf.” https://ebpf.io/what-is-ebpf/#what-is-ebpf, 2023. Accessed: 2023-10-12.