

# Cyber Threat Modeling of an LLM-Based Healthcare System

Neha Nagaraja<sup>1</sup><sup>a</sup> and Hayretin Bahsi<sup>1,2</sup> <sup>b</sup>

<sup>1</sup>*School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, U.S.A.*

<sup>2</sup>*Department of Software Science, Tallinn University of Technology, Tallinn, Estonia*

**Keywords:** Healthcare, Large Language Models, Cyber Threats, Conversational Attacks, Adversarial Attacks, Threat Modeling.

**Abstract:** With the rapid advancement of large language models (LLMs) and their integration into the healthcare system, it is critical to understand their resiliency against cyber-attacks since sensitive data handling is paramount. Threat modeling is most important, as addressing cybersecurity early in system development is essential for safe and reliable deployment. While traditional threat modeling practices are well-established, applying these frameworks to systems integrating LLM, especially in healthcare, presents unique challenges. It is essential to examine conventional cyber threats, adversarial threats, and threats specific to LLM in tandem to build robust defense mechanisms. This paper adapts the STRIDE methodology to assess threats in LLM-powered healthcare systems holistically, identifying components and their data flows and mapping potential threats introduced by each component. It provides practical guidance for understanding the threats early in development and demonstrates effective system modeling tailored to healthcare settings.


## 1 INTRODUCTION


Large Language Models (LLMs), such as GPT-4 (OpenAI), LLaMA (Meta), and Bard (Google), have revolutionized natural language processing by setting new benchmarks in generating human-like text, driving advancements in fields such as conversational agents, content automation, and data analysis for decision-making systems (Chu et al., 2024)(Shah et al., 2024). The widespread accessibility of these models via user-friendly platforms, such as chatbots and APIs, has enabled various industries to leverage sophisticated language processing without the need for specialized expertise (Abbasian et al., 2024)(Deng et al., 2024). In healthcare, LLMs have significantly enhanced medical information retrieval, patient communication, diagnostic support, and personalized treatment recommendations (Abbasian et al., 2024)(Tang et al., 2024).

However, integrating LLMs into healthcare systems introduces significant ethical and cybersecurity challenges that must be carefully addressed (Pankajakshan et al., 2024). Recent studies have highlighted various threats associated with LLMs, including prompt injection attacks, jailbreak methods, data

contamination, and the generation of misinformation, all of which exacerbate these risks and necessitate advanced defense mechanisms (Chu et al., 2024)(Pathmanathan et al., 2024)(Balloccu et al., 2024). These threats can pose significant risks, including the spread of false medical advice, breaches of sensitive patient data, and a loss of trust in healthcare systems (Brown et al., 2024)(Greshake et al., 2023). The potential for LLMs to generate and propagate misinformation adds another layer of complexity, particularly in healthcare, where accurate information is paramount (Shah et al., 2024). While LLMs offer transformative potential for the healthcare sector (Abbasian et al., 2024), their integration must be accompanied by robust threat modeling and comprehensive security measures. Threat modeling is a crucial early-stage activity in the secure development lifecycle (SDLC), enabling the identification and assessment of potential risks. By systematically categorizing and addressing threats, we can implement adequate safeguards that protect both the integrity of healthcare systems and the well-being of patients.

Attacks on the conversational interfaces of LLMs (e.g., prompt injection, jailbreaks) have recently drawn significant interest within research communities. LLMs can also be targeted by attacks (e.g., model inversion, model extraction) inherited from the

<sup>a</sup> <https://orcid.org/0009-0000-2967-244X>

<sup>b</sup> <https://orcid.org/0000-0001-8882-4095>

adversarial machine learning security domain. As LLMs are part of integrated systems in real-world implementations, threat actors can resort to conventional cyber threats (e.g., eavesdropping, denial of service) against the attack surfaces. Threat actors can combine various threat types in a single attack campaign. Thus, it is imperative to have a holistic understanding of all three threat landscapes in a unified threat modeling framework. Such a perspective has not been reflected in existing threat modeling studies.

This paper demonstrates how the widely used STRIDE threat modeling approach can be adapted to systems with LLM-based components. Specifically, we illustrate an LLM-based system using a data flow diagram (DFD) for a healthcare system case study. We then identify security boundaries and systematically elicit threats relevant to the system, focusing on adversarial, conversational, and conventional threats. The unique contribution lies in our detailed demonstration of system modeling for the LLM-based case study and systematic threat elicitation, integrating both the LLM and cybersecurity perspectives.

Despite the importance of threat modeling, there is a notable scarcity of research focusing on LLM-based systems. This is especially true for studies addressing system modeling and systematic threat elicitation. Effective threat modeling for LLM-integrated healthcare systems necessitates combining conventional, adversarial, and LLM-specific conversational threats. Our paper addresses this research gap by offering a structured approach to threat modeling that considers the unique challenges posed by LLMs in healthcare. By doing so, we provide a valuable framework for securing LLM-integrated systems, an area that remains underexplored in the current literature.

The paper is organized as follows: Section 2 reviews related work, Section 3 outlines the study's methodology, Section 4 presents the case study results, Section 5 discusses key findings, and Section 6 concludes the paper.

## 2 RELATED WORK

Research on large language model security and robustness has rapidly advanced, uncovering a wide range of vulnerabilities that threaten their safe usage across various domains. Attacks such as prompt injections—jailbreaks have been a focus as adversaries manipulate inputs to elicit malicious outputs from the models. Many studies have classified and comprehensively evaluated how these attacks are applied so that even minimal manipulations can induce harmful behaviors (Chu et al., 2024)(Deng et al., 2024)(Gre-

shake et al., 2023)(Rossi et al., 2024). In addition, the integration of third-party LLM plugins and APIs in the application raises new vulnerabilities, such as proprietary information leakage and malicious exploitation (Iqbal et al., 2024)(Zhao et al., 2024)(Finlayson et al., 2024). Data contamination is a critical threat resulting from the manipulation of training data, leading to degraded model performance and enabling the extraction of proprietary knowledge without access to the original dataset (Pathmanathan et al., 2024)(Balloccu et al., 2024)(Truong et al., 2021).

In healthcare, LLMs are used for tasks like medical knowledge retrieval and personalization of treatments (Abbasian et al., 2024)(Tang et al., 2024), but their strengths also make them easy to misuse, such as spreading false medical advice, misinformation, or unauthorized access to sensitive patient data (Shah et al., 2024)(Chen and Shu, 2024). Studies have shown that sophisticated attack vectors like indirect prompt injection can remotely compromise LLM-integrated healthcare applications, leading to severe consequences such as data breaches and the undermining of trust in medical systems (Chu et al., 2024)(Greshake et al., 2023).

To mitigate these threats, frameworks such as STRIDE (Tete, 2024) and risk assessments (Pankajakshan et al., 2024) are used to identify and evaluate risks. However, despite using STRIDE & DREAD, Tete's framework focuses on high-level threat identification without linking threats to specific system components (Tete, 2024). Similarly, Pankajakshan et al. emphasize broad risk categorization across stakeholders but lack detailed, component-specific threat elicitation (Pankajakshan et al., 2024). Both approaches fail to provide the structured system modeling needed to identify which components introduce specific threats, limiting their effectiveness for targeted security measures.

The categorization of LLM threats is provided by resources such as MITRE ATLAS (MITRE, 2024), the OWASP Top 10 LLM threats (OWASP, 2024), and NIST Adversarial ML (Vassilev et al., 2023). These frameworks highlight the importance of aligning LLMs with human values & adhering to regulatory standards, such as the European Union's AI Act and the NIST AI Risk Management Framework, to ensure ethical & secure deployment (Tang et al., 2024). These resources are instrumental knowledge bases for threat modeling.

Our paper uniquely advances the field by focusing on structured system modeling and systematic threat elicitation, pinpointing which components generate specific threats. Unlike prior studies, we comprehensively address common cyberattack vectors, adversar-

ial threats, and conversational threats within a unified threat assessment framework, providing a holistic, actionable guide for secure LLM deployment.

### 3 METHODS

In this section, we outline the approach taken to identify and address potential security threats within our LLM-based healthcare system. The objectives employed in this study are defined in the security objectives. We next defined our system and its components in the system modeling phase and then conducted a comprehensive threat elicitation. By establishing a clear attack taxonomy, we aim to systematically categorize potential threats for each component.

#### 3.1 Security Objectives

Our study aims to identify potential threats in a healthcare system integrated with large language models (LLMs). We are focused on protecting traditional and LLM-specific threats to ensure the system's CIA Triad—confidentiality, integrity, and availability.

We have categorized threats into three main types:

(1) Conventional Cyber Threats — These are typical cyber threats that mainly target the infrastructure and core system components, compromising the confidentiality, integrity, and availability of the systems, such as MiTM attacks, unauthorized access, malware, or denial of service. In our system, unauthorized access to healthcare data & patient information is critical due to the sensitive nature of the data and the importance of regulatory compliance (e.g., HIPAA).

(2) Adversarial Threats — These threats specifically target the machine learning models (Papernot et al., 2018). As our system incorporates different models to provide decision-making support, these threats can compromise the behavior and performance of the models. These attacks include inference attacks, model extraction, model evasion, and so on (Vassilev et al., 2023). In our system, attackers might first exploit traditional cyber threats to access critical assets (e.g. bypassing web applications, firewalls, or APIs) before executing adverse attacks on the models.

(3) Conversational Threats — These threats specifically target interactions between user input and external sources that feed into the LLM. Such threats compromise the integrity of LLM outputs by manipulating the model's operation or behavior. We introduce this additional category to distinguish between general adversarial threats and the more specific attacks that exploit conversational interfaces. These attacks often manifest through direct or indirect prompt

injections (OWASP, 2024).

Given this context, our specific security assumptions include:

1) In this paper, we rely on third-party LLMs. We do not have direct control over their training process or data assets. Therefore, the risk of training data poisoning attacks falls within the scope of the Service-level agreements (SLAs) with third-party providers and is not a concern in this study. Instead, we focus on risks that we can manage and control.

2) Fine-tuning of the LLMs is considered outside the scope of our current objectives. Hence, any risks like data leakage or adversarial manipulation are eliminated. Our focus is solely on the secure use of pre-trained models provided by third-party vendors.

3) We aim to uphold the intellectual property rights of the third-party LLMs used in our system.

In the scope of the study, we do not provide complete privacy threat modeling. Instead, we focus on the security threat model of the LLM-based system architecture. Although our model identifies various privacy risks, it does not address all privacy concerns, such as linkability or identifiability. Privacy-centric analysis requires frameworks similar to LINDDUN (Deng et al., 2011).

#### 3.2 System Modeling

We perform system modeling to better understand the complex interactions within a system. In this study, we have adopted a Level 1 Data Flow Diagram (DFD) to model the system, breaking down the process into sub-processes and illustrating data flows and stores.

There are three ways of incorporating LLMs into any system (I. S. Authority, 2024) - (1) Integrating LLM via API provided by an external provider (OpenAI, Anthropic). (2) Importing a pre-trained LLM (open-source or third-party like Mistral) with an option of further fine-tuning. (3) The organization designs, trains, and deploys its own LLM (in-house LLM). In this study, we follow the second approach by using a pre-trained model without fine-tuning.

To assess potential threats, we have utilized the STRIDE framework (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege), which is applied to the DFD elements as shown in the table below. The threats were mainly identified based on the attack taxonomy.

We designed our DFD based on a conversational health agent called openCHA, an open-source framework with a large language model (LLM) accessed via API (Abbasian et al., 2024). However, in our study, we have opted for Option 2—importing a pre-trained LLM from a third-party or open-source

Table 1: STRIDE-per-Element: Applicable Threats to DFD Elements (Shostack, 2014).

DFD Element	S	T	R	I	D	E
Entity	✓		✓			
Process	✓	✓	✓	✓	✓	✓
Data Flow		✓		✓	✓	
Data Store		✓	✓	✓	✓	

provider (e.g., Mistral) to have better control over data. We do not fine-tune the model; instead, we use it as-is and deploy it within our infrastructure. By doing so, we ensure that sensitive healthcare data is processed securely without relying on external API calls from third-party providers.

The system is divided into five major components: Web Application, Healthcare Platform, Orchestrator, External Sources and LLM, as illustrated in Figure 1.

In the openCHA, as described by Abbasian et al. (2024), the Orchestrator consists of the Task Executor, Task Planner, Prompt List, Response Generator, and Data Pipe. For simplicity, we have combined the roles of the Prompt List with the Task Executor and the Response Generator with the Task Planner.

Unlike the original openCHA framework, where Healthcare Platforms were treated as external resources, we've chosen to integrate the Healthcare Platform directly into our system. This is because it holds crucial patient information that needs to remain secure and cannot be exposed to outside entities. By including it within the system boundary, we can better protect sensitive patient data and ensure that it is handled with care. Using these data, we can generate a more personalized response.

To effectively manage and maintain the LLM, we have divided its life cycle (IBM, 2024) (Shi et al., 2024) into three primary stages: The first stage involves model deployment and versioning, ensuring seamless integration and updates. The second stage is the LLM process, which handles queries and generates responses, followed by the third stage, which focuses on performance metrics, monitoring, and logging to ensure accuracy, efficiency, and compliance.

While designing our DFD diagram, we were determined to depict each component accurately. When deciding whether to represent the LLM component as a data store or a process, we considered several factors - (1) Its role in the system. (2) The level of control we have over it, (3) how well it fits within the architecture. Since the LLM plays an active role in processing user inputs, generating strategies, and interacting with components like the Task Planner combined with the level of control, we have over LLM's operations and determined that LLM functions as a process rather than a data store.

From the STRIDE perspective, if we treat LLM as a data store, the primary threats would be T, R, I, and D. However, it is essential to account for S and E, as LLM could be spoofed by an attacker impersonating all legitimate entries, leading to malicious or incorrect inputs being processed. If the system misconfigures access controls, the LLM could gain elevated privileges, allowing unauthorized access to sensitive data or critical system actions.

We have represented the third-party repository as an entity in our system because it is an external resource over which we have no direct control. Our main concerns are Spoofing (ensuring we receive updates from a legitimate source) and Repudiation (verifying and tracking the receipt of updates). T, I, D, and E threats against these repositories are not concerns in our study (handled by SLAs) as we do not alter, access sensitive information, or manage the repository's operations. However, the information flow between the external resource and our system is subject to various threats (e.g., eavesdropping or MiTM) that we consider in this study.

The data flow from the Third-Party Repository (entity) to the Model Deployment & Versioning (process) represents periodic updates or new versions received by our system. The third-party repository is external to our system and we have no direct control over it. All version information and updates are stored in the data store called the model store. It is essential to continuously track and evaluate LLM performance, such as speed and resource utilization, and track logs to identify bottlenecks, ensure transparency, and provide audit material, which is extremely important in healthcare. This is done using the Performance metrics and Monitoring process shown in our DFD. These performance data and inference logs are stored in the Metrics & Logs store (data store).

In Section 4.1, we will dive deeper into the system's data flows, explaining how each component works together.

### 3.3 Security Boundaries

In STRIDE, security or trust boundaries are defined as the borders between different trusted zones within the system. They are crucial for identifying areas where potential security risks may arise. The boundaries are drawn based on the system's attack surface to reflect where potential malicious interactions could occur. During threat elicitation, any components defined within a trust boundary are considered trusted and generally excluded from direct analysis. All data flows that cross these boundaries are suspected and must be thoroughly analyzed as they present a higher



risk. More details on the security boundaries and their impact on our Threat Modeling process are explained in Section 4.2.

### 3.4 Threat Elicitation

Before we delve into threat elicitation, it is crucial to establish a comprehensive attack taxonomy. This will provide a clear framework for understanding the various potential threats and serve as a foundation for the subsequent analysis.

**Attack Taxonomy.** We have utilized the MITRE ATLAS framework in our attack taxonomy, extending it with specific threats from the OWASP Top 10 for LLM applications and traditional cybersecurity threats. This comprehensive approach ensures that general cybersecurity and LLM-specific threats are addressed, which is especially crucial in all domains, including healthcare.

The mapping between OWASP and MITRE ATLAS demonstrates that several OWASP-identified threats align directly with corresponding MITRE ATLAS entries. For instance, LLM01 (Prompt Injection) corresponds to MITRE ATLAS's LLM Prompt Injection, LLM03 (Training Data Poisoning) maps to Poisoning Training Data, LLM04 (Model Denial of Service) to Denial of ML Service, LLM05 (Supply Chain Vulnerabilities) to ML Supply Chain Compromise, LLM06 (Sensitive Information Disclosure) to LLM Data Leakage, LLM07 (Insecure Plugin Design) to LLM Plugin Compromise, and LLM10 (Model Theft) to Extract ML Model (MITRE, 2024) (OWASP, 2024). However, some OWASP threats, specifically LLM02 (Insecure Output Handling), LLM08 (Excessive Agency), and LLM09 (Overreliance) do not have direct equivalents in the MITRE ATLAS framework.

Of these, LLM09: Overreliance is considered out of scope for our study since we are not focusing on the hallucination of LLMs, and we assume the LLM outputs to be trustworthy. Therefore, to create a comprehensive attack taxonomy for our system, we have extended the MITRE ATLAS framework to include two additional threats from OWASP: LLM02: Insecure Output Handling and LLM08: Excessive Agency. In addition, we have considered traditional cybersecurity threats to grasp the complete threat modeling. We acknowledge that as the number of LLM applications in healthcare grows, new vulnerabilities may develop, demanding constant updates.

In this study, we use STRIDE-per-element rather than STRIDE-per-interaction to systematically identify threats for each system component, considering specific attack methods, preconditions, and mitigations (Shostack, 2014). This structured approach im-

proves clarity in mapping threats to DFD components, such as entities, data stores, processes, and data flows. In section 4.3, we will delve deeper into the specifics of our threat modeling process, providing a detailed analysis of how we applied the STRIDE-per-element methodology across different components of our healthcare system.

## 4 RESULTS

### 4.1 System Modeling

The DFD in Figure 1 describes the entire system. We identified five entities, thirty-three data flows, seven processes, and four data stores. The User (i.e., patient or any healthcare provider such as a doctor) is considered an entity in our system. We have divided the system into five major components:

**Web Application:** The Web application serves as the user interface, allowing users to input natural language queries and ensuring secure data transmission between the user and the system.

**Healthcare Platform:** This (process) stores sensitive patient health records in the Patient Database (data store). It interacts with the Web Application and the Task Executor to provide patient data, which can be either presented to the User or used for a personalized response.

**Orchestrator:** As described by Abbasian et al. (2024), the Orchestrator is one of the system's core components. We divided it into Task Executor (process); Task Planner (process); and Data Pipeline (data store). The Task Executor is responsible for converting and processing user queries and extracting the necessary information; it also interacts with external resources. It plays a central role in managing the workflow within the system. It sends user queries and relevant data to the Task Planner to generate strategies and responses. It also requests the Healthcare Provider and gets the patient data sent to the Task Planner for a personalized response. It interacts with External Resources to get translated text clinical knowledge data and ai/ml analyzed patient insights. The Task Planner is the decision-making component of the system. It analyzes user queries, formulates a strategy, and divides the tasks required to generate a response. It works with the LLM to refine the strategy and develop a personalized response. It also sends the relevant instructions to the Task Executor for execution. All intermediate data from these instructions and analyzed patient insights and user prompts are stored in the Data Pipeline which is a repository that manages metadata from external sources and data gener-

ated during task execution. This information is crucial in multi-step process thinking and is available to the Task Executor and the Task Planner.

**External Resources:** As described by Abbasian et al. (2024), these are external entities that provide the system with additional data, insights, knowledge, and translation capabilities. We have considered a Translator, Knowledge Base, and AI & Analysis Models as our external resources for this system. Each of these resources is accessed via API by the Task Executor. If a user's inputs are non-English, the Task Executor sends a translation request to the Translator, which converts the query into English, supporting multiple languages in the system. The Knowledge Base retrieves up-to-date information from trusted healthcare sources to prevent bias or hallucination in LLM responses. The AI & Analysis Models leverage analytics tools to derive insights, complementing LLMs to enhance data processing and inference capabilities.

**Large Language Model Component:** This core component is responsible for processing natural language input, understanding context, and generating appropriate responses based on user queries and healthcare data. We have further divided this into three lifecycle stages (IBM, 2024) (Shi et al., 2024) of the LLM. First is Model Deployment & Versioning, which is represented as a process responsible for the deployment and versioning of models, ensuring the system uses the latest and most secure versions. A Third-party Repository, identified as an entity, provides all these model versions stored in the Model Store, which is considered a data store. The data flow between the deployment process and the repository is infrequent and periodic, as it periodically retrieves model weights and updates from the repository to keep the LLM current. The Model Store also maintains version information, ensuring that the correct model is used for processing queries. Second is the LLM model, which receives structured prompts and data from the Task Planner to generate strategies and responses. These responses are then sent back for further processing, continuing in a bidirectional data flow until a satisfactory outcome is achieved. Lastly, Performance and Monitoring (process) monitors the system's performance, logs data, and maintains records to ensure the LLM's accuracy, efficiency, and compliance. It collects all relevant information from the LLM and stores it in the Metrics & Logs Store, which is identified as a data store. This step is crucial for monitoring the health of the data to maintain optimal system performance and reliability.

## 4.2 Security Boundaries

We have identified and defined eight distinct security boundaries within our DFD, which are described in Table 2.

## 4.3 Threat Elicitation

In this section, we conduct a detailed threat elicitation for each component of our system, utilizing the STRIDE framework in conjunction with the attack taxonomy outlined in the methods section. Due to page limitations, we will analyze a single entity, data flow, process, and data store from the five entities, thirty-three data flows, seven processes, and four data stores identified.

**Entity.** For this analysis, we have chosen the User entity:

**Spoofing:** Hijacking User Accounts (Iqbal et al., 2024) occurs when attackers exploit weak authentication or stolen credentials to access sensitive patient data. Hijacking User Machines (Iqbal et al., 2024) involves using malware or phishing to compromise devices and infiltrate hospital networks. Credential Stuffing happens when attackers use reused passwords from previous breaches to gain unauthorized access (MITRE, 2024).

**Repudiation:** Lack of Audit Trails results from inadequate logging and weak access controls, allowing malicious actions to go undetected, such as unauthorized alterations of patient records. Manipulation of Logs occurs when attackers alter or delete log entries to conceal their activities, making breaches difficult to trace. User Impersonation exploits weak authentication and shared accounts to perform actions as another user, enabling denial of these actions. Unauthorized Actions on Behalf of Users arise from insufficient authentication and authorization, allowing actions like unauthorized prescriptions. Insider Threats involve employees misusing legitimate access to steal data or perform unauthorized actions, undermining patient confidentiality and trust.

**Data Flow.** For this analysis, we have chosen User to Web Application data flow:

**Tampering:** Prompt Injection (Jailbreak) (LLM01) exploits vulnerabilities in LLM-based systems by accepting unsanitized user inputs due to inadequate input validation and weak model constraints (MITRE, 2024). Attackers can be registered users with malicious intent or individuals using compromised accounts who have access to the user interface and understand the parameter structures, enabling them to manipulate input fields or API parameters (Chu et al., 2024). These manipulations can

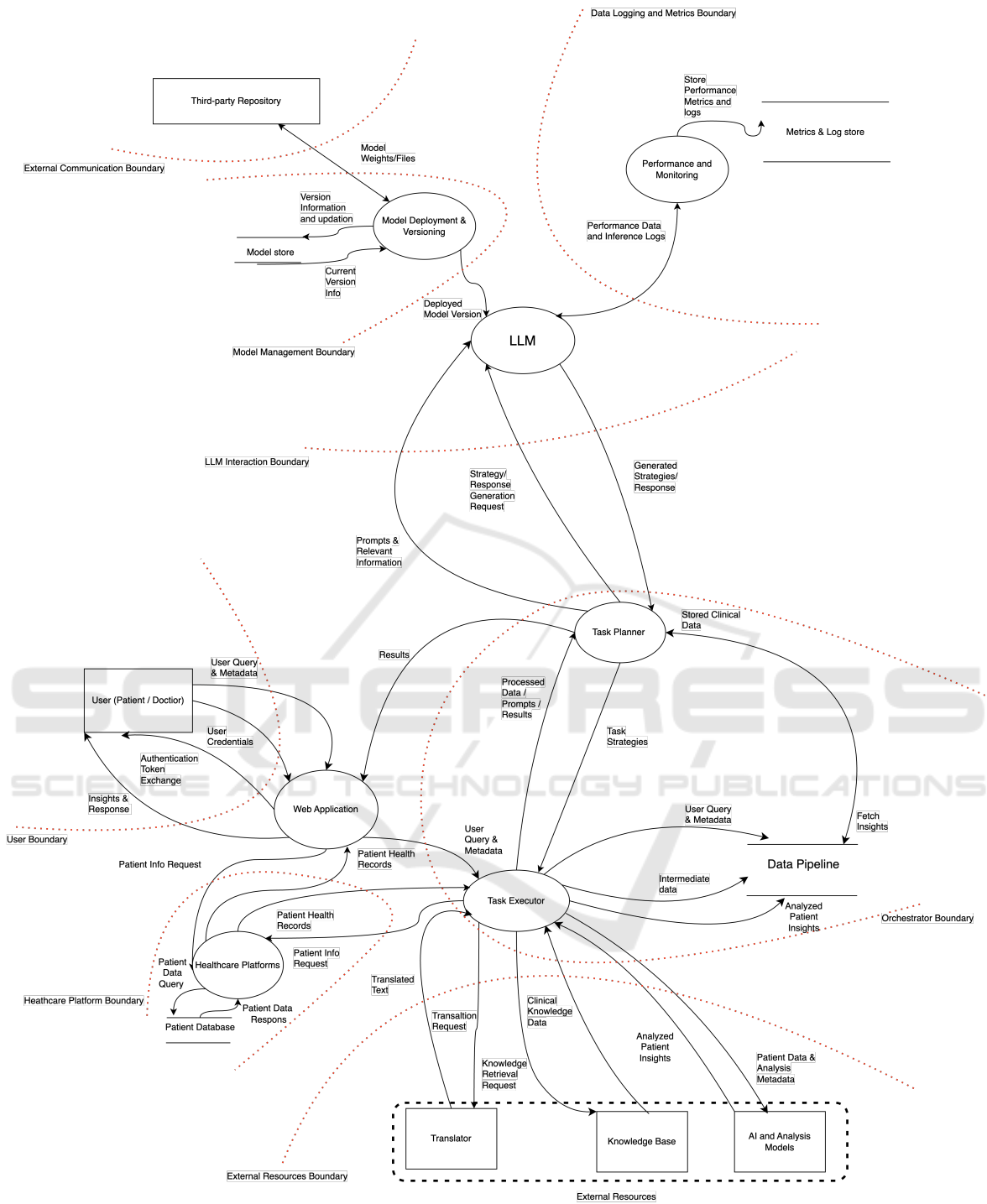


Figure 1: Data Flow Diagram.

lead to unauthorized access to sensitive information, data breaches, and the generation of misleading or harmful outputs, thereby compromising the overall security and integrity of the system.

There are different methods used in this attack

(Chu et al., 2024): Attackers use role-play or style injection to bypass restrictions (Human-Based). Utilizing optimization algorithms to refine prompts and elicit restricted outputs (Optimization-Based). Attackers use encoded or obfuscated prompts to by-

Table 2: Security Boundaries.

Boundary Name	Components Inside Boundary	Trust Assumptions	Data Flows Crossing Boundary	Explanation
User Boundary	User (Patient/Doctor)	Web Application does not trust input from external users.	User queries and requests entering the Web Application.	Defines the entry point for external user inputs, which are untrusted and need monitoring.
Healthcare Platform Boundary	Healthcare Platform, Patient Database	Components within this boundary trust each other.	Data requests and queries from Web Application and Task Executor.	Encompasses components that store and manage sensitive patient data, all of which are trusted internally and secured with best practices.
External Resources Boundary	Translator, Knowledge Base, AI and Analysis Models	External resources are untrusted.	Interactions between Orchestrator and External Resources.	Includes third-party tools, focusing on untrusted data exchanges that could be exploited.
Orchestrator Boundary	Task Executor, Task Planner, Data Pipeline	Components within this boundary trust each other.	Data exchanges with External Resources, Web Application, Healthcare Platform, and LLM.	Groups key orchestrator components, emphasizing cautious handling of untrusted interactions at its edges.
LLM Interaction Boundary	LLM	Data crossing this boundary cannot be trusted.	Inputs to and outputs from the LLM.	Identifies interactions with the LLM as potential attack surfaces due to untrusted data exchanges.
External Communication Boundary	Third-party Repository	Third-party Repository is untrusted.	Interactions between Model Deployment & Versioning and Third-party Repository.	Highlights the risk of introducing malicious code through external model updates.
Model Management Boundary	Model Deployment & Versioning, Model Store	Components within this boundary trust each other.	Data flows from Third-party Repository or LLM.	Encompasses components managing the LLM, assuming internal trust while treating external data flows as potential attack surfaces.
Data Logging and Metrics Boundary	Performance and Monitoring, Metrics & Logs Store	Components within this boundary trust each other.	Interactions between LLM and Performance and Monitoring process.	Includes components for logging and monitoring, with data exchanges across this boundary considered untrusted and vulnerable.

pass content filters (Obfuscation-Based). Modifying request parameters to alter backend behavior or the LLM’s output (Parameter-Based). MITM attacks, as outlined in Table 3, are within this scope, alongside Replay, HTTP Response Splitting, and XML Injection attacks.

**Information Disclosure: Man-in-the-Middle (MITM) Attacks** can also expose sensitive information like patient symptoms and medical histories. Eavesdropping exploits unencrypted data transmissions and insecure communication protocols, allowing attackers to silently capture sensitive data such as patient IDs and treatment plans.

**Sensitive Information Disclosure (LLM06) (OWASP, 2024)** occurs when LLMs is trained on sensitive data without adequate safeguards and weak data privacy controls as LLM memorize sensitive information from its training data. Attackers can craft specific queries that exploit this capability and retrieve confidential data. For example, an attacker might query, "Provide an example patient record similar to 'John Doe' with epilepsy," causing the LLM to disclose sensitive patient details (Rossi et al., 2024).

**Denial of Service: Flooding/DDoS Attack** targets

the limited request-handling capacity of web application interfaces lacking traffic filtering. In healthcare, this could involve flooding the interface with numerous simultaneous queries, preventing doctors and patients from accessing critical medical advice or patient records.

**Data Store.** For this analysis, we have chosen the Data pipeline datastore:

**Tampering: Data Manipulation Attacks** exploit weak access controls and insufficient data integrity checks to alter stored data, such as modifying patient metadata, which leads to incorrect analyses and compromised patient care. **Metadata Injection Attacks** take advantage of inadequate input validation to insert or modify metadata, misleading the system and potentially causing misdiagnoses or inappropriate treatment plans.

**Repudiation:** Similar to the case of the user entity, manipulation or deletion of logs and insufficient audit trails allow attackers to erase or alter records, hiding malicious activities like incorrect data insertions in the Data Pipeline datastore. In healthcare, this can lead to undetected tampering with patient data, compromising data integrity and regulatory compliance.

**Information Disclosure: Sensitive Data Exposure**



results from inadequate access controls and weak encryption, allowing attackers to access unencrypted patient data. Unauthorized Data Access arises from weak access control mechanisms, enabling attackers with compromised credentials to manipulate or delete critical health data.

**Denial of Service: Flooding/DDoS Attacks on the Data Pipeline** Datastore overload the system with excessive requests. Unlike attacks on web interfaces, which disrupt access to services, these target the data store directly, hindering data retrieval and updates. Resource Exhaustion via Complex Queries consumes significant processing power, delaying access to essential healthcare data on time.

**Process.** For this analysis, we have chosen the LLM process:

**Spoofing:** Model version spoofing occurs when an attacker delivers fake model updates by manipulating communication channels or versioning requests, causing the LLM to provide faulty medical advice and endangering patients. Task Planner spoofing happens when an attacker impersonates the Task Planner due to a lack of mutual authentication, enabling them to send malicious prompts that result in fraudulent responses and disrupt patient care.

**Tampering: Prompt Injection (LLM01) in LLM - Tampering** threats in the LLM process, such as Direct Prompt Injection Jailbreak Attacks, exploit insufficient input sanitization, allowing attackers to inject harmful instructions. In healthcare, this could lead to the LLM issuing unsafe medical advice. Similar methods outlined in the data flow section for jailbreaking are applied here as well.

**Indirect Prompt Injection** occurs when attackers manipulate external data sources to inject harmful prompts (Greshake et al., 2023). This is due to the system's dependence on external data and the lack of robust input filtering. Attackers use various methods (Greshake et al., 2023) such as passive methods, where malicious prompts are embedded in public data sources or code repositories. Active methods, where attackers deliver prompts directly, such as via emails processed by the systems. User-driven Injections exploit social engineering, tricking users into copying and pasting malicious prompts unknowingly. Finally, Hidden Injections uses obfuscation techniques like multi-stage exploits or encoded prompts, making detection harder by embedding prompts in images or using encrypted payloads through executable scripts. In healthcare, for example, an attacker could manipulate an external knowledge base response, resulting in inaccurate medical recommendations from the LLM.

**Repudiation:** Attackers manipulate LLM logs to conceal the injection of malicious prompts, prevent-

ing the detection of incorrect patient treatment suggestions. Denial of Query Submission occurs when inadequate tracking of query history allows attackers to modify logs and deny responsibility for submitting malicious queries, resulting in the LLM generating incorrect medical advice without traceability and posing significant risks to patient safety.

**Information Disclosure: Sensitive Data Exposure (LLM06)** (OWASP, 2024) is a concern here. In the context of LLMs, attackers can access sensitive patient data or even training data processed by the model, leading to breaches of patient confidentiality and potential misuse of proprietary information. This can be achieved through methods like prompt injection, where an attacker crafts inputs to bypass input filters and convince the LLM to reveal confidential data that it has memorized or processed.

**Insecure Output Handling (LLM02)** occurs when LLM outputs are integrated into external web interfaces without proper sanitization (OWASP, 2024). If an attacker has access to the API or can manipulate web output, they can inject malicious scripts or code, such as cross-site scripting (XSS) or cross-site request forgery (CSRF), through LLM-generated responses (OWASP, 2024). An attacker injects a script via LLM output in a patient portal, exploiting XSS to access sensitive health data.

**Denial of Service: Model Denial of Service (LLM04)** occurs when attackers flood the LLM with a large number of queries, causing it to crash and preventing doctors from receiving real-time medical guidance. They also send complex medical queries that consume significant LLM resources, resulting in delayed responses for critical patient care. Additionally, overwhelming the LLM with multiple simultaneous tasks further delays response times and impacts timely patient treatment.

**Elevation of Privilege: Privilege Escalation through Model Vulnerabilities** occurs when weak access controls, such as inadequate RBAC or poorly secured API keys, allow attackers to modify model parameters or access sensitive data. In a healthcare context, this can enable unauthorized access to patient data or alteration of treatment algorithms, compromising patient care and privacy.

A selection of attacks is detailed in Table 3, located after the References section.

## 5 DISCUSSION

In this paper, we deployed a third-party LLM (Commercial Off-The-Shelf, or COTS) into our system. Had we continued to use the LLM as an API, as in the

referenced system (Abbasian et al., 2024), both system modeling and threat evaluation would have been different. We would have considered the LLM API as an entity in our DFD, with the primary threats identified being related to impersonation (Spoofing) and lack of accountability (Repudiation). The advantages are that it is easier to set up the system to use LLM API, and it ensures scalability and that the system is up to date. However, this also comes with its own risks, such as DNS spoofing or API Key theft, which could allow attackers to set up a fake endpoint and impersonate the LLM API, which could lead to data manipulation and unauthorized access. Having inadequate logging or log manipulation can make verifying actions or detecting malicious activities difficult. Since we deployed the LLM model into our system, we can ensure that sensitive information remains in our secure environment, as there can be vulnerabilities associated with data transmission over external networks. Ultimately, the decision on how LLM is adapted into the system is based on the organization's priorities and risk tolerance.

Prompt injection attacks are identified in multiple components of our system - the User-to-Web Application data flow and the LLM process. This dual elicitation arises because such attacks exploit the vulnerabilities at different system points. Countermeasures can, therefore, be implemented at various levels: input filters can be deployed in the data flow between the user and the web application to sanitize incoming prompts, while fine-tuning the LLM can enhance its ability to reject improper prompts internally. This approach underscores the importance of collaboration between cybersecurity (handling input filtering) and data science (responsible for model development). These teams often operate independently, which can hinder the implementation of comprehensive security measures. Recognizing and addressing these interdependencies is crucial during countermeasure analysis, as it allows for a more effective and cost-efficient defense strategy against prompt injection attacks.

One of the limitations of our study is that we did not address the threat of training data poisoning, as our approach does not involve training or fine-tuning the models internally. We expect the third-party providers to maintain security while training the model through Service Level Agreements (SLAs). While the threat of training data poisoning is significant as it can potentially undermine the reliability and trustworthiness of the LLM output it remains beyond the scope of our current work. By acknowledging this limitation underscores the need for future research to explore strategies for mitigating such risks.

Additionally, we opted for the STRIDE-per-

element approach instead of STRIDE-per-interaction in our threat elicitation process (Shostack, 2014). This decision allows us to more easily associate specific threats with individual system components, simplifying the process for the technical team and making it more accessible to stakeholders who may not have extensive technical expertise. Using STRIDE-per-element also supports detailed system modeling. We can more accurately identify and represent components like LLMs, offering a clearer view of potential vulnerabilities specific to each part of the system.

## 6 CONCLUSION & FUTURE WORK

In this paper, we conducted a comprehensive threat modeling analysis of a healthcare system powered by large language models (LLMs). By adapting the STRIDE methodology, we created a system model and systematic approach to identify component-level threats and map potential vulnerabilities. In the future, we aim to establish tailored countermeasures to enhance resilience.

## 7 DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the revision of this work, the author(s) used ChatGPT (OpenAI's chat-based AI) for grammar and language review. The author(s) subsequently reviewed and edited the content as needed and take full responsibility for the publication's final content.

## REFERENCES

- Abbasian, M., Azimi, I., Rahmani, A. M., and Jain, R. (2024). Conversational health agents: A personalized llm-powered agent framework.
- Balloccu, S., Schmidtová, P., Lango, M., and Dušek, O. (2024). Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms.
- Brown, H., Lin, L., Kawaguchi, K., and Shieh, M. (2024). Self-evaluation as a defense against adversarial attacks on llms.
- Chen, C. and Shu, K. (2024). Can llm-generated misinformation be detected?
- Chu, J., Liu, Y., Yang, Z., Shen, X., Backes, M., and Zhang, Y. (2024). Comprehensive assessment of jailbreak attacks against llms.

- Deng, G., Liu, Y., Li, Y., Wang, K., Zhang, Y., Li, Z., Wang, H., Zhang, T., and Liu, Y. (2024). Masterkey: Automated jailbreaking of large language model chatbots. In *Proceedings 2024 Network and Distributed System Security Symposium*. Internet Society.
- Deng, M., Wuyts, K., Scandariato, R., Preneel, B., and Joosen, W. (2011). A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32.
- Finlayson, M., Ren, X., and Swayamdipta, S. (2024). Logits of api-protected llms leak proprietary information.
- Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. (2023). Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection.
- I. S. Authority (2024). Tehisintellekti ja Masinõppe Tehnoloogia Riskide ja Nende Leevendamise Võimaluste Uuring.
- IBM (2024). What Are Large Language Model Operations (LLMOps)? Accessed: 2024-11-13.
- Iqbal, U., Kohno, T., and Roesner, F. (2024). Llm platform security: Applying a systematic evaluation framework to openai's chatgpt plugins.
- MITRE (2024). Atlas matrix. Accessed: Oct 21, 2024.
- OWASP (2024). Owasp top 10 for large language model applications. Accessed: Oct 21, 2024.
- Pankajakshan, R., Biswal, S., Govindarajulu, Y., and Gressel, G. (2024). Mapping llm security landscapes: A comprehensive stakeholder risk assessment proposal.
- Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. (2018). SoK: Security and Privacy in Machine Learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE.
- Pathmanathan, P., Chakraborty, S., Liu, X., Liang, Y., and Huang, F. (2024). Is poisoning a real threat to llm alignment? maybe more so than you think.
- Rossi, S., Michel, A. M., Mukkamala, R. R., and Thatcher, J. B. (2024). An early categorization of prompt injection attacks on large language models.
- Shah, S. B., Thapa, S., Acharya, A., Rauniyar, K., Poudel, S., Jain, S., Masood, A., and Naseem, U. (2024). Navigating the web of disinformation and misinformation: Large language models as double-edged swords. *IEEE Access*.
- Shi, C., Liang, P., Wu, Y., Zhan, T., and Jin, Z. (2024). Maximizing User Experience with LLMOps-Driven Personalized Recommendation Systems.
- Shostack, A. (2014). *Threat Modeling: Designing for Security*. John Wiley & Sons.
- Tang, X., Jin, Q., Zhu, K., Yuan, T., Zhang, Y., Zhou, W., Qu, M., Zhao, Y., Tang, J., Zhang, Z., Cohan, A., Lu, Z., and Gerstein, M. (2024). Prioritizing safeguarding over autonomy: Risks of llm agents for science.
- Tete, S. B. (2024). Threat modelling and risk analysis for large language model (llm)-powered applications.
- Truong, J. B., Maini, P., Walls, R. J., and Papernot, N. (2021). Data-free model extraction.
- Vassilev, A., Oprea, A., Fordyce, A., and Anderson, H. (2023). *NIST Trustworthy and Responsible AI NIST AI 100-2e2023 Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations*. NIST. Available at <https://doi.org/10.6028/NIST.AI.100-2e2023>, Accessed Oct 21, 2024.
- Zhao, W., Khazanchi, V., Xing, H., He, X., Xu, Q., and Lane, N. D. (2024). Attacks on third-party apis of large language models.

## APPENDIX

Table 3: Threat Elicitation Table (selected samples). Type - Conventional Cyber Threats: CCT, Adversarial Threats: AdvT, Conversational Threats: ConT DFD Elements - E: Entity, DF: Data Flow, P: Process, DS: Data Store.

Type	DFD Element	Identified Threats	Threat Description
AdvT	User to Web Application (DF), LLM (P)	Sensitive Information Disclosure (LLM06)	Exploitation of LLM memory to retrieve confidential data by crafting specific queries (OWASP, 2024) (MITRE, 2024).
AdvT	LLM (P)	Model Denial of Service (LLM04)	An attacker overloads resource usage or manipulates the context window to degrade LLM performance and increase operational costs (OWASP, 2024) (MITRE, 2024).
AdvT	LLM (P)	Model Version Spoofing	Attackers deliver fake model updates, causing compromised model behavior and inaccurate healthcare advice.
AdvT	LLM (P)	Privilege Escalation through Model Vulnerabilities	Weak access control and poor API security allow attackers to access and alter sensitive model parameters.
AdvT	LLM (P)	Insecure Output Handling (LLM02)	Poor sanitization of LLM outputs allows attackers to inject malicious scripts, risking unauthorized access to sensitive data
ConT	User to Web Application (DF), LLM (P)	Prompt Injection (Jail-break)	Attackers exploit input validation weaknesses to manipulate prompts, potentially generating unsafe medical advice (OWASP, 2024)
ConT	LLM (P)	Resource Exhaustion via Prompt Manipulation	Attackers overwhelm the system by crafting computationally intensive prompts, causing service degradation.
ConT	LLM (P)	Indirect Prompt Injection	Manipulation of external sources allows attackers to inject harmful prompts, generating inaccurate responses (OWASP, 2024) (MITRE, 2024).
CCT	User (E)	Browser Session Hijacking	Exploiting weak session management, attackers intercept session tokens to impersonate users and access sensitive data.
CCT	User (E), Data Pipeline (DS), LLM (P)	Manipulation of Logs	Attackers alter or delete log files to conceal malicious activities and evade detection.
CCT	User to Web Application (DF)	Man-in-the-Middle (MITM) Attack	Attackers intercept and alter unencrypted communication, exposing sensitive data and potentially misguiding treatment.
CCT	Data Pipeline (DS)	Data Manipulation Attack	Attackers alter stored data due to weak access controls, impacting data integrity and healthcare decisions.
CCT	Data Pipeline (DS), LLM (P)	Sensitive Data Exposure	Inadequate encryption and access control lead to unauthorized access to sensitive patient data.
CCT	User to Web Application (DF)	Exploiting Vulnerabilities for Denial of Service	Attackers exploit unpatched vulnerabilities in the web application, causing system crashes and unavailability of healthcare resources.