

Facilitating Threat Modeling by Leveraging Large Language Models

Isra Elsharef
University of Wisconsin-Milwaukee
isra@uwm.edu

Zhen Zeng
University of Wisconsin-Milwaukee
zhenzeng@uwm.edu

Zhongshu Gu
IBM Research
zgu@us.ibm.com

Abstract—In recent years, security engineers in product teams have faced new challenges in threat modeling due to the increasing complexity of product design and the evolving nature of threats. First, security engineers must possess a thorough understanding of how to translate the abstract categories of threat modeling methodology into specific security threats relevant to a particular aspect of product design. Without such in-depth knowledge, applying threat modeling in practice becomes a difficult task. Second, security engineers must be aware of current vulnerabilities and be able to quickly recall those that may be relevant to the product design. Therefore, for effective threat modeling, a deep understanding of a product’s design and the background knowledge to connect real-time security events with specific design principles embedded in large volumes of technical specifications is required. This can result in a lot of human effort invested in parsing, searching, and understanding what is being built through design documents and what threats are relevant based on that information. We observe that the recent emergence of large language models (LLMs) may significantly change the landscape of threat modeling by automating and accelerating the process with their language understanding and logical reasoning capabilities. In this paper, we have developed a novel LLM-based threat modeling system by leveraging NLP techniques and an open-source LLM to decrease the required human effort above in the threat modeling process. In the evaluation, two major questions of threat modeling (MQ1 and MQ2) are considered in the proposed workflow of Task 1 and Task 2, where the NLP techniques assist in parsing and understanding design documents and threats, and the LLM analyzes and synthesizes volumes of documentation to generate responses to related threat modeling questions. Our initial findings reveal that over 75% of the responses meet the expectations of human evaluation. The Retrieval Augmented Generation (RAG)-enhanced LLM outperforms the base LLM in both tasks by responding more concisely and containing more meaningful information. This study explores a novel approach to threat modeling and demonstrates the practical applicability of LLM-assisted threat modeling.

I. INTRODUCTION

Threat modeling is a critical component throughout a product’s lifecycle and plays an important role in ensuring its security [19], [28]. The OWASP Foundation, a nonprofit that tries to improve the security of software through its community-

led open-source software projects, defines threat modeling as a process to identify, communicate, and understand threats & mitigations within the context of protecting something of value [20]. A working group of experienced security professionals created the Threat Modeling Manifesto, which defines four major questions (MQs) used as a guide during the threat modeling process [9]:

- MQ1: What are we working on?
- MQ2: What can go wrong?
- MQ3: What are we going to do about it?
- MQ4: Did we do a good enough job?

These four MQs are used to identify and address security threats in a system [20]. In practice, questions can be further broken down into more specific and detailed queries, each tailored to correspond with the specific characteristics of the given input files. Answering these detailed queries can require diving into large amounts of documents and communication across different teams to extract relevant information [22]. As a result, threat modeling is commonly viewed as a demanding and lengthy process in practice [19].

Threat modeling is typically part of the product development process throughout the key stages of product design, development, and testing and involves communication between multiple roles, i.e., security engineers and product architects. Figure 1 illustrates this. The product team may reference documentation initially for context when creating the threat model. The product security engineer is engaged whenever relevant documentation is ready for review, and the product architect clarifies and updates documentation and the threat model based on threat modeling discussions [19]. Security engineers must possess a thorough understanding of how to translate the abstract categories of threat modeling methodology into specific security threats relevant to a particular piece of product design. Without such in-depth knowledge, applying threat modeling effectively in practice becomes a difficult task [17].

In addition, the complexity and difficulty of such tasks are further amplified by the constantly expanding threat landscape and the escalating complexity in product system design [22], [19]. The National Vulnerability Database (NVD) [5] is the U.S. government repository of standards-based vulnerability management data, which enables vulnerability management, security measurement, and compliance. The NVD has a

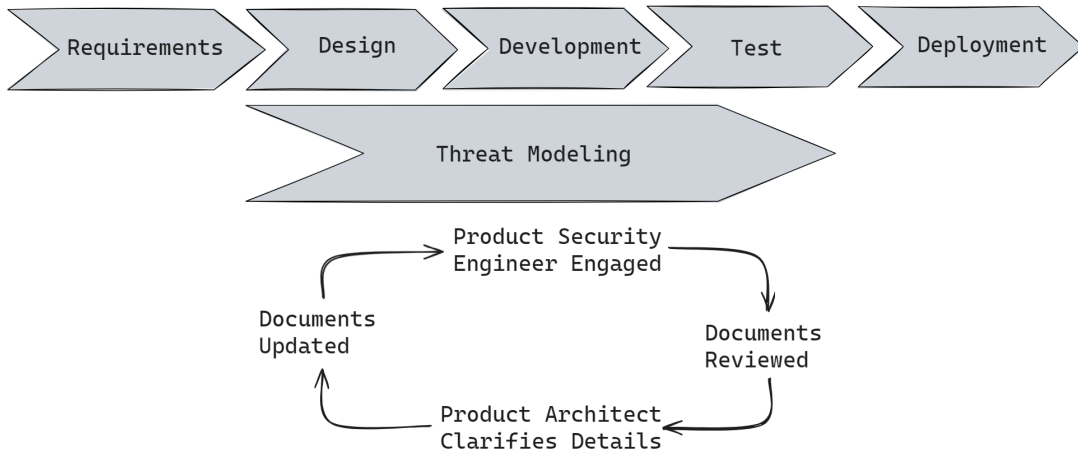


Fig. 1. Threat Modeling in Product Development Process

non-static nature and is continuously updated. For example, approximately 27,000 vulnerabilities were added throughout 2023. The rapid increase of new vulnerabilities can be overwhelming for security engineers to keep up to date with, and it's a challenge to monitor and analyze new security threats so they can be accurately considered in practice when threat modeling.

This study focuses on developing a large language model (LLM)-based threat modeling system to facilitate answering the first two major questions MQ1 and MQ2. The question *MQ1: What are we working on* is about understanding the system and application being developed, while *MQ2: What can go wrong* involves identifying potential security threats and vulnerabilities in a system. In this threat modeling system design, we develop the proposed system using tools like KeyBERT [4] for NLP tasks and an open-source LLM Llama 2 [29]. This study demonstrates how the proposed LLM-based threat modeling generated answers for MQ1 in Task 1 (in Section III-C) and for MQ2 in Task 2 (in Section III-D). For evaluation, detailed questions were defined for MQ1 and MQ2 based on the contents of input files. Six questions were asked of each threat model input document for a total of 72 question queries. Questions in Task 1 focused on the area of system and design. For Task 2, questions were more security-related, focusing on secrets management, identity & access control, and network configurations. The results of this study reveal that over 75% of the responses meet the expectations of human evaluation. The Retrieval Augmented Generation (RAG)-enhanced LLM outperformed the base LLM in both tasks, responding with more concise and meaningful information.

This paper is organized into the following sections: Section II summarizes the related work of threat modeling, Section III illustrates the system and model design of the proposed LLM-based threat modeling, Section IV explains the evaluation and results in this study, Section V discusses limitations and future work, and Section VI is the conclusion.

II. RELATED WORK

Threat modeling is an analytical process focused on examining a system's architecture or design to identify and mitigate security vulnerabilities [19]. The STRIDE framework is the most common approach to creating threat models and targets what could go wrong [8]. Existing threat modeling frameworks like STRIDE are used as guidance to build threat models, which include details like data flow, trust boundaries, and application usage scenarios [30]. Identifying these details can be a slow and painful process that often involves digging through multiple types of documentation [22].

In practice, threat modeling is highly recommended to be implemented during the architecture or design phase. Engaging in threat modeling at these early stages allows for proactively identifying potential security vulnerabilities [19]. This begins with identifying and categorizing the assets that need protection, followed by mapping out the potential attack vectors and threat agents that could exploit system vulnerabilities [19]. The process involves scrutinizing the system architecture to understand how different components interact and where weaknesses may exist. Regularly revisiting and updating the threat model is crucial, especially as new threats emerge and the product evolves [30]. This comprehensive analysis not only aids in developing robust security measures tailored to the product's specific needs but also ensures that the development team maintains a security-focused mindset throughout the product's lifecycle, leading to a more secure and resilient final product [19], [22], [30].

The predominant tool for threat modeling is the Microsoft Threat Modeling Tool [26]. This tool was initially released in 2014 and can be considered one of the more mature tools available. Some tools are diagram-based in that end users drag and drop to draw system diagrams, ultimately leading to a generated summary of potential threats based on applied rules from a library. Other tools are text-based, where models are expressed in a structured language. Some weaknesses identified are related to duplicate threats and missing threat information (such as mitigation and initial recommended priority) in the

generated reports. Another consideration that is not taken into account with these tools is the larger context around certain types of systems, such as those built for the Internet of Things (IoT) [27], [18].

A recent study reveals that the practical application of threat modeling methodologies encounters various challenges, ranging from updating asset identification and data flow diagrams to cross-product modeling [17]. Another challenge is information can be unstructured and spread across multiple documents, increasing the time required to generate an accurate threat model. This was exemplified in this recent study, where even in a best-case scenario, 45 hours were spent on threat model creation [30].

III. SYSTEM DESIGN

In this study, we propose an LLM-based threat modeling approach. Figure 2 shows an overview of the proposed system design, which contains functions for extracting key characteristics from input data, constructing a knowledge base, and generating answers to the detailed questions of each MQ. This study focuses on two tasks of addressing MQ1 and MQ2 in threat modeling:

- Task 1: Understanding the system and application, discussed in Section III-C.
- Task 2: Identifying potential security threats, discussed in Section III-D.

The proposed approach facilitates the existing threat modeling process by automatically extracting key product design characteristics from given documentation and answering threat modeling questions for security engineers. This reduces the need for security engineers to manually comb through documents from the beginning and streamlines the threat identification process.

A. Data

In this study, we consider two types of input data. The first is design documents, which define the system design and architecture and address its security, scalability, and performance aspects. The second is vulnerability data, which is from the NVD. There are over 231,000 vulnerabilities in the NVD. Each vulnerability is associated with a unique ID, a Common Vulnerabilities and Exposures (CVE) identifier. Each CVE has fields for descriptions, severity, references, weaknesses, and configuration. Severity is based on the Common Vulnerability Scoring System (CVSS), and CVSS vector strings include exploitability and impact metrics that range from 1-10, with 10 being the most severe. The exploitability score represents Access Vector, Access Complexity, and Authentication. Each CVE vulnerability is accompanied by a description, which provides details about the vulnerability, including how it can be exploited, its potential impact, and how it can be mitigated or resolved. This information could help security engineers understand the threat posed by a vulnerability.

B. Large Language Model

In this study, we explore LLM-assisted threat modeling based on the model of Llama 2 [29]. Meta’s Llama 2 family of open-source LLMs, released in July 2023, was selected because it generally performs better than existing open-source models based on testing the Meta team did using a series of helpfulness and safety benchmarks. It also appears on par with closed-source models when they performed human evaluations [29]. Additionally, Llama emphasizes safety as seen by the released details for how safety was prioritized during pre-training and fine-tuning, the use of red teaming (by internal employees, contract workers, and external vendors), and the use of human evaluation to judge safety violations. In this study, we utilize the pre-trained 7B & 13B Llama 2-CHAT models, fined-tuned versions of Llama 2 that are optimized for dialogue use cases and were run in Google Colab using T4 GPU.

We also utilize the retrieval-augmented generation (RAG) AI framework to give the most accurate and up-to-date information [12]. The RAG model utilizes a pre-trained language model Llama 2 and integrates it with a retriever component, allowing it to access external knowledge sources for context adaption [25]. Implementing RAG models requires technologies such as vector databases, which allow for the rapid coding of new data and searches against that data to feed into the LLM [11]. In traditional databases, querying is done for rows in the database where the value usually exactly matches the query. In vector databases, a similarity metric is applied to find a vector that is the most similar to the query.

C. Data Processing and Knowledge Base in Task 1

Task 1 aims to support security engineers in understanding the system and application. Design documents are the inputs. The developed system processes the input data by following the four steps below before using the large language model.

- 1-1: Load and extract text from input data (e.g., PDF document). Split and chunk text by size.
- 1-2: Create embeddings (vectors used to compress size) for each text chunk.
- 1-3: Create Pinecone index; an index is the highest-level organizational unit of vector data in Pinecone [10].
- 1-4: Push data to Pinecone vector database to make it accessible by LLM model.

LangChain [7] was used to load multiple related documents simultaneously. LangChain is a framework for developing applications powered by language models. A significant benefit of using LangChain is that it can be a standard interface for interacting with different models. For one of the tasks defined in this research, the loaded document has metadata fields for “source” specifying the document name and page number. For the other task, the metadata has a field “text” for vulnerability descriptions and “pdf.keyword” which is the keyword used to identify the relevant vulnerability based on the loaded PDFs. When splitting text, the RecursiveCharacterTextSplitter was used, and the parameters `chunk_size` and `chunk_overlap` were

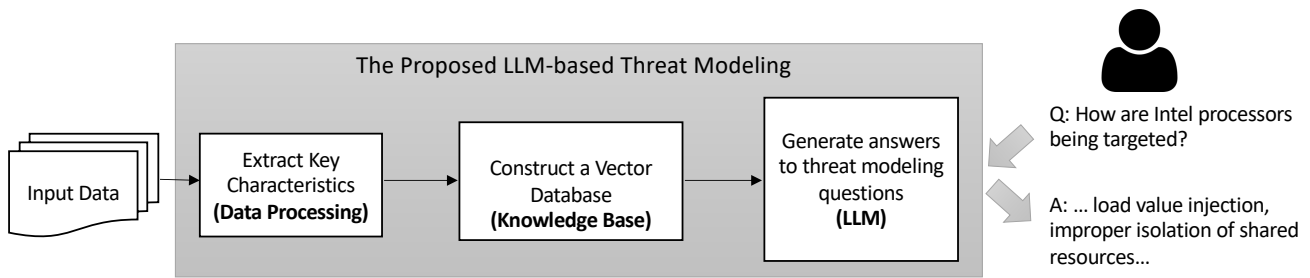


Fig. 2. Overview of Proposed LLM-based Threat Modeling

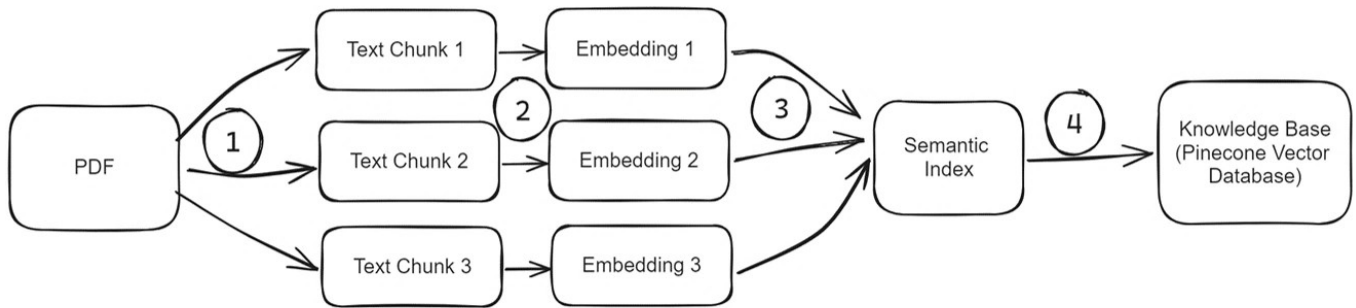


Fig. 3. Workflow of Data Processing and Knowledge Base in Task 1

specified. The `chunk_size` was set to 500 characters, and the `chunk_overlap` was 20. This text splitter is parameterized by a list of characters, the default being `["\n\n", "\n", " ", ""]`, and tries to split text to the specified length [7].

D. Data Processing and Knowledge Base in Task 2

Task 2 aims to support security engineers in identifying potential security threats, where the input data could be design documents or vulnerability data. The main workflow of data before being used by the LLM is shown in Figure 4, which contains four steps as follows:

- 2-1: Keywords extracted from documentation as a CSV file.
- 2-2: NVD API programmatically queried with extracted keywords; Relevant fields extracted from response to generate a CSV file.
- 2-3: Data preprocessed to get in the required format for the model
- 2-4: JSONL file specified as input to be accessible to the LLM via Pinecone vector database

In Task 2, we extracted keywords from the design documentation PDF files. Each PDF was read, and the keywords were extracted using NLP techniques. English stopwords and expert-defined stopwords were removed. The remaining keywords and “expert-identified” keywords were then used to search the NVD via API and retrieve CVEs with their descriptions. That data was then further processed to for the required format expected by the model. KeyBERT [4] was used to extract key characteristics of design documentation. KeyBERT depends on Bidirectional Encoder Representations from Transformers

(BERT)-embeddings and cosine similarity to find the keywords and key phrases in a document that are most similar to the document itself [4]. Embeddings are numerical representations of pieces of information, such as text or documents, that capture semantic meaning. Using KeyBERT enables extracting relevant keywords that efficiently capture semantic information from the input text.

TABLE I
THE EXAMPLE OF REMOVED STOPWORDS

Stopword Type	Example
NLTK English Stopwords	start, but, to, and, the, in, it, are, I, that,...
Expert-defined Stopwords	allow, attack, hacker, security, cyberattack, cyber, cybersecurity, intelligence, architecture, secure, ...

When using KeyBERT, two types of stopwords are removed: the default “English” stopwords as defined by the NLTK (Natural Language Toolkit) [6] and “expert-defined” stopwords as shown in Table I. Stopwords are commonly used words in a language and are filtered out before processing in NLP tasks. When creating the “expert-defined” stopword list, we focused on words typically associated with security but so widely used that they would not be useful representations of keywords. KeyBERT also allows the parameters “`top_n`” and “`keyphrase_ngram_range`” to be specified. The value of “`top_n`” was set to 15. Setting a range allows for key phrases, which could help more with context. Stemming via NLTK was also used to avoid running keyword searches that returned duplicate results.

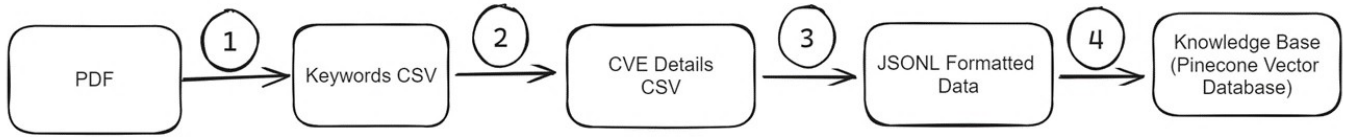


Fig. 4. Workflow of Data Processing and Knowledge Base in Task 2

In constructing the customized vulnerability dataset for Task 2, Python scripts that go through the steps below were developed. These scripts support constructing a vulnerability dataset automatically and ensure that the constructed vulnerability dataset is always up-to-date by having the capability to query the NVD with identified keywords to pull the latest vulnerabilities.

The vulnerability knowledge base is formatted as a JSONL (JSON Lines) file. Each entry in the JSONL file corresponds to a vulnerability CVE that was returned from querying the NVD with a keyword. The associated fields for the CVE can be set as metadata when making the information available to the LLM. The main benefits of using a JSONL file are stream processing where data can be read and written incrementally, error handling such that an error in one entry does not make the entire file unusable and space efficiency because commas or other separates are not needed between objects [13]. Pinecone [10] was specified as the vector store type in this experiment. The index_name, embeddings, and field in metadata for the text chunks that contained text content were passed in. The vulnerability knowledge base is stored as the Pinecone Vector Database.

IV. EVALUATION AND RESULTS

To evaluate the developed system for Task 1 and Task 2 in Section III, two sets of questions for MQ1 and MQ2 were designed. Twelve design documents were used as input during evaluation. The PDFs ranged from three pages to over sixty pages. All documents were in English and included figures, and most included tables. The documents were from multiple sources, including IEEE Xplore [2], Cybersecurity & Infrastructure Security Agency site [1], and Intel Trust Domain Extensions (TDX) documentation [3]. The PDFs were treated as independent of one another because of the availability of related documentation. When documents are loaded, the LangChain loader points to a directory, so any PDF included in that directory is loaded. The metadata ties the extracted text to that document and page number.

We designed a set of detailed questions for each input file in Task 1 and Task 2, and 6 questions were asked of each threat model of input for a total of 72 queries. Table II shows example questions used for evaluation. Specifically for the system design and architectures, questions in Task 1 contain the area of system description, system environment, and data. For Task 2, these questions cover different cybersecurity areas, including secrets management, identity & access control, network configurations, and working scenarios. The number of entries in the custom vulnerability dataset ranged from 1,000

to over 36,000, depending on the input file, where the total number of CVEs in the NVD is over 231,000.

TABLE II
THE EXAMPLE OF THREAT MODELING QUESTIONS IN TASK 1 AND 2

Tasks	Area	Questions
Task 1	System description	What components are involved in the design?
Task 1	System environment	How is the encrypted connection established?
Task 1	Data	How is data secured during transmission?
Task 2	Secrets management	How can shared private keys used for authentication be exposed?
Task 2	Identity & access control	What are some concerns related to open source authentication methods?
Task 2	Network	Other than a hash function and timestamp, what can be used to avoid a replay attack?

To evaluate the performance of the proposed framework of LLM-based threat modeling, results are compared between the base LLM with and without Retrieval Augmented Generation (RAG). Human evaluation was adopted to evaluate the output of answers to questions in Task 1 and Task 2 in this study. The method of human evaluation is essential in determining how well the model can meet user expectations and needs. Human evaluation has been widely used to evaluate the performance of LLM [16]. In this study, we use “+” to represent an accepted output that meets the evaluator’s expectations and “-” for answers that do not meet expectations. In the human evaluation processes, two coders code these outputs independently and consolidate all codes for generating the final evaluation. The overall satisfaction rate is over 75%. For example, Table III shows an example of human evaluation codes for positive and negative answers. This negative evaluation is because the response was too general and without enough valuable information.

We also counted how long it took to receive a response for each query. The query response times ranged from 25 to 50 seconds. This variation in response time can be affected by several factors, including the way data is stored and the specific configurations of the Pinecone database for these tasks. Additionally, the network connection between Google CoLab and Pinecone might be critical in these observed time frames. Further investigation is needed to identify acceptable response times and understand what contributes to these longer-than-desired times.

In addition, to better understand the LLM model’s performance, we compared the system’s output for base LLM and RAG-enhanced LLM. Table IV shows an example for the

TABLE III
THE EXAMPLES OF OUTPUT EVALUATION

Question	Response	Eval
How is the encrypted connection established?	Based on the provided context, the encrypted connection between an Aggregator and an EV is established using the following steps: 1. The Aggregator signs a message ...3. When an EV decides to sell electricity to an Aggregator, it uses the Diffie-Hellman key exchange to establish ... Therefore, the answer to the question is: The encrypted connection is established using the Diffie-Hellman key exchange and shared private keys for data encryption and decryption in successive connections.	+
What are examples of similarities between vulnerabilities found IoT, IT, and OT networks?	All three types of networks have vulnerabilities related to improper authentication and information disclosure.	-

question of *How are intel processors being targeted?* for both base LLM and RAG-enhanced LLM. This question aims to provide more context when looking at a design document related to Intel Trust Domain Extensions (Intel TDX), which can be especially useful for situations where either the product security engineer or the development team may not be experienced or may not have had much exposure to certain types of threats. As shown in Table IV, the RAG-enhanced LLM produced responses that were not only more concise but also more precise, often incorporating specific and relevant details due to the vector database. The base LLM generates lengthier content that can be irrelevant or include hallucinations. In this example, RAG-enhanced LLM responses were more concise and sometimes included descriptions of relevant vulnerabilities from the vector database generated in the knowledge base stage. In this study, the RAG-based LLM generated an average of 74 words compared to the 253 words for the base LLM.

TABLE IV
THE EXAMPLE OF OUTPUT IN DIFFERENT LLM SETTINGS

LLMs	Response
Base LLM	Intel processors have been the subject of several high-profile security vulnerabilities ... some ways that Intel processors have been targeted: 1. Spectre and Meltdown: These two vulnerabilities were discovered in 2018 and affect nearly all modern processors, including those from Intel. They allow an attacker to access sensitive information, such as passwords and encryption keys, by exploiting 2. Side-channel attacks ... such as cryptographic keys or other confidential data. 3. Firmware vulnerabilities ... 4. Supply chain attacks ... 5. Physical attacks ... To protect against these types of attacks, it is important to keep your devices and software up to date ...
RAG-enhanced LLM	Based on the information provided, it appears that Intel processors are being targeted through various vulnerabilities, including load value injection, improper isolation of shared resources, and sequence of processor instructions leading to unexpected behavior. These vulnerabilities can be exploited by authenticated users with local access to potentially enable information disclosure, escalation of privilege, and denial of service attacks.

More interestingly, the RAG-enhanced LLM model showed the capability of the “awareness of honesty” in instances where questions were not supported by sufficient information in the input documentation, for example, the response that

Based on the provided context, I don't know the answer to your question. The passage does not provide any information about how the custom Modbus TCP/IP protocol stack is implemented. The authors of the passage seem to be focused on the use of Modbus TCP in industrial processes and the challenges of simulating Modbus communication in Simulink, rather than detailing the implementation of the custom protocol stack. Therefore, I cannot provide an answer to your question.

In this example, the RAG-enhanced LLM communicated its inability to respond accurately because of a lack of information.

In another case, shown in Table V, the input files contain sufficient information, but the response to the initial question does not provide enough valuable information. By modifying the question prompt with more specified details, e.g., the connection between MSKs and MTKs in this example, the output response contains more detailed information related to the question. However, in this example, we observed the inaccurate response for *MTKs (Migration Trust Anchors)*, which should be *MTKs (Migration Transport Keys)*. When the RAG-enhanced LLM model was tasked with summarizing the system design and architecture components in a query, it usually returned more accurate and relevant responses than the base LLM in this study.

TABLE V
THE EXAMPLE RESPONSE FOR TASK 2 WHEN PROMPT REQUIRED MODIFICATION

Prompt	Response
How are MSKs used and protected?	Based on the context provided, MSKs (Migration Security Keys) are used and protected as follows: MSKs are used after the MigTD policy verification. This suggests that MSKs are used as part of the migration process, specifically after the policy verification step. As for how MSKs are protected, the context does not provide any information. Therefore, I cannot answer this question.
How are MSKs used and how are they protected using MTKs?	Based on the context provided, MSKs (Migration Security Keys) are used and protected using MTKs (Migration Trust Anchors) . MSKs are passed after the MigTD policy verification, which suggests that MSKs are used to establish a secure communication channel between these systems or applications, ... MTKs, on the other hand, are used to protect MSKs by providing a trusted anchor for the migration process. MTKs are used to verify the authenticity of the MSKs and ensure that they are being used correctly. This helps to prevent unauthorized access to the data being transmitted and ensures that the communication channel is secure. In summary, MSKs are used to establish a secure communication channel between different systems or applications, while MTKs are used to protect MSKs by providing a trusted anchor for the migration process.

V. DISCUSSION

In this study, we developed an LLM-based threat modeling system to address the challenge of threat modeling in answering two major questions MQ1 and MQ2. Although we observed promising output responses generated by the developed system, there are some limitations to using LLM for threat modeling.

Large language models can offer benefits in simplifying the threat modeling process, but they also possess weaknesses that could impact the quality of their outputs. A critical concern is the potential for biases stemming from their training datasets. Research has been done comparing models that have significant guardrails, such as the OpenAI GPT models vs open source models without guardrails [14]. Findings indicate that those without guardrails can be more easily biased towards less common viewpoints when given prompts intentionally designed to influence that way. Another concern is about hallucinations, which refer to incorrect responses generated by LLMs. Such incorrect responses may result from biased datasets, a lack of real-world knowledge, or an inadequate understanding of input. Efforts to reduce the frequency of hallucinated responses are ongoing, with one approach involving the use of guided iterative prompting [23]. Table V shows the example of hallucination from RAG-based LLM responses. Existing studies with larger datasets have shown that RAG-enhanced LLMs return fewer hallucinations than LLMs without RAG. In some cases, hallucinations can be useful, such as for creating an agent for testing, but to avoid missing critical tests, this use case should still be augmented with expert review [21].

Additionally, because of the sensitive nature of the documentation and the potential to identify weaknesses in products, the privacy of any shared information is required. Using LLMs for threat models built across teams can result in privacy risk stemming from membership inference and training data extraction [15]. A significant barrier to scaling LLMs is the cost and time to train [24]. This originally meant access to models was limited to large companies and research labs, but recent efforts, such as the release of the Llama 2 collection of models and the integration available with cloud platforms, have been made that allow training and deploying LLMs to be more accessible.

Further work can be considered to enhance the effectiveness and applicability of this RAG-enhance LLM solution for threat modeling. Based on the observations in this study, the RAG-enhanced LLM model shows the capability of handling threat modeling-related queries effectively. The technique challenges are rooted in investigating how to improve the query response accuracy and time.

VI. CONCLUSION

This study developed a novel LLM-based threat modeling approach to decrease the required human effort in the threat modeling process. NLP techniques were employed to assist in parsing and understanding design documents and threats, while an open-source LLM was used to analyze and synthesize

documentation. Two major threat modeling questions around understanding the system and application, and identifying potential security threats were targeted in this study. By demonstrating the use of LLMs to facilitate threat modeling within the proposed proof-of-concept framework, this study's findings reveal an increase in the efficiency of threat modeling by allowing development and security teams to utilize natural language queries to understand the product design and quickly identify potentially relevant vulnerabilities.

REFERENCES

- [1] Cybersecurity & infrastructure security agency. <https://www.cisa.gov/>.
- [2] IEEE Xplore. <https://ieeexplore.ieee.org/>.
- [3] Intel Trust Domain Extensions. <https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/documentation.html>.
- [4] KeyBERT. <https://maartengr.github.io/KeyBERT/>.
- [5] National vulnerability database. <https://nvd.nist.gov/>.
- [6] Natural language toolkit. <https://www.nltk.org/>.
- [7] Recursively split by character langchain. <https://python.langchain.com/>.
- [8] Stride (security). <https://en.wikipedia.org/>.
- [9] Threat Modeling Manifesto. <https://www.threatmodelingmanifesto.org/>.
- [10] Understanding indexes. <https://docs.pinecone.io/docs/indexes>.
- [11] What is retrieval augmented generation (RAG)? <https://www.oracle.com/artificial-intelligence/generative-ai/retrieval-augmented-generation-rag/>.
- [12] What is retrieval-augmented generation?, February 2021. <https://research.ibm.com/blog/retrieval-augmented-generation-RAG>.
- [13] You Need To Start Using JSONL In Your Golang Projects!, July 2023. <https://thegodev.com/handling-jsonl/>.
- [14] Nicklaus Badyal, Derek Jacoby, and Yvonne Coady. Intentional biases in llm responses. In *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0502–0506. IEEE, 2023.
- [15] Hannah Brown, Katherine Lee, Fatemehsadat Miresghallah, Reza Shokri, and Florian Tramèr. What does it mean for a language model to preserve privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 2280–2292, 2022.
- [16] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*, 2023.
- [17] Daniela Soares Cruzes, Martin Gilje Jaatun, Karin Bernsmed, and Inger Anne Tøndel. Challenges and experiences with applying microsoft threat modeling in agile development projects. In *2018 25th Australasian Software Engineering Conference (ASWEC)*, pages 111–120. IEEE, 2018.
- [18] Lucas Peixoto da Silva, Bernardo S Nascimento, Rafael A Mello P Dias, and Diogo S Mendonça. A comprehensive approach for applying threat modeling to internet of things systems. In *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, pages 01–06. IEEE, 2022.
- [19] Danny Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy*, 9(4):41–47, 2011.
- [20] V. Drake. Threat modeling. https://owasp.org/www-community/Threat_Modeling.
- [21] Robert Feldt, Sungmin Kang, Juyeon Yoon, and Shin Yoo. Towards autonomous testing agents via conversational large language models. *arXiv preprint arXiv:2306.05152*, 2023.
- [22] Jeffrey A Ingalsbe, Louis Kunimatsu, Tim Baeten, and Nancy R Mead. Threat modeling: diving into the deep end. *IEEE software*, 25(1):28–34, 2008.
- [23] Susmit Jha, Sumit Kumar Jha, Patrick Lincoln, Nathaniel D Bastian, Alvaro Velasquez, and Sandeep Neema. Dehallucinating large language models using formal methods guided iterative prompting. In *2023 IEEE International Conference on Assured Autonomy (ICAA)*, pages 149–152. IEEE, 2023.
- [24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

- [25] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [26] Microsoft. Microsoft threat modeling tool overview. <https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool>.
- [27] Amar Seeam, Ochanya S Ogbah, Shivanand Guness, and Xavier Bellekens. Threat modeling and security issues for the internet of things. In *2019 conference on next generation computing applications (NextComp)*, pages 1–8. IEEE, 2019.
- [28] Peter Torr. Demystifying the threat modeling process. *IEEE Security & Privacy*, 3(5):66–70, 2005.
- [29] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [30] Koen Yskout, Thomas Heyman, Dimitri Van Landuyt, Laurens Sion, Kim Wuyts, and Wouter Joosen. Threat modeling: from infancy to maturity. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, pages 9–12, 2020.