

LARGE LANGUAGE MODEL ASSISTED THREAT MODELING

by

Isra Elsharef

A Thesis Submitted in  
Partial Fulfillment of the  
Requirements for the Degree of

Master of Science  
in Computer Science

at

The University of Wisconsin-Milwaukee

December 2023

## ABSTRACT

### LARGE LANGUAGE MODEL ASSISTED THREAT MODELING

by

Isra Elsharef

The University of Wisconsin-Milwaukee, 2023  
Under the Supervision of Professor Zhen Zeng, Ph.D.

Threat modeling aims to identify and address potential threats early in the product development lifecycle, but is often a time-consuming process involving extensive collaboration between product security and development teams, and relying heavily on analyzing various input documentation. This thesis explores the use of Retrieval Augmented Generation (RAG) Large Language Models (LLMs) as an innovative approach to enhance the threat modeling process. This study is pioneering in its use of LLMs for this purpose and the creation of a subset of related vulnerabilities to be passed as input to make sure the model has access to up-to-date information. The findings of this study reveal the capability of utilizing a RAG LLM to assist in threat modeling.

© Copyright by Isra Elsharef, 2023  
All Rights Reserved

## TABLE OF CONTENTS

LIST OF FIGURES . . . . .	v
LIST OF TABLES . . . . .	vi
ACKNOWLEDGEMENTS . . . . .	vii
<b>1 Introduction . . . . .</b>	<b>1</b>
<b>2 Background . . . . .</b>	<b>2</b>
2.1 <b>Threat Modeling Process . . . . .</b>	<b>2</b>
2.2 <b>Connected Vehicle Vulnerabilities . . . . .</b>	<b>5</b>
<b>3 Methodology . . . . .</b>	<b>6</b>
3.1 <b>Model . . . . .</b>	<b>7</b>
3.2 <b>Data . . . . .</b>	<b>11</b>
<b>4 Results . . . . .</b>	<b>20</b>
4.1 <b>Evaluation . . . . .</b>	<b>22</b>
4.2 <b>Analysis . . . . .</b>	<b>41</b>
4.2.1 <b>Observations made for Task 1 . . . . .</b>	<b>41</b>
4.2.2 <b>Observations made for Task 2 . . . . .</b>	<b>42</b>
4.3 <b>Limitations . . . . .</b>	<b>45</b>
<b>5 Future Work . . . . .</b>	<b>46</b>
<b>6 Conclusion . . . . .</b>	<b>46</b>
REFERENCES . . . . .	47

## LIST OF FIGURES

1	Example threat modeling diagram in the Microsoft Threat Modeling tool [4] . . . . .	4
2	Threat modeling as part of development process . . . . .	5
3	Overview of connected vehicle vulnerabilities [8] . . . . .	6
4	RAG pipeline [15] . . . . .	9
5	Common pipeline for a vector database [15] . . . . .	9
6	Pinecone architecture [17] . . . . .	10
7	RAG application with LangChain [18] . . . . .	11
8	NVD dashboard [21] . . . . .	12
9	NVD vulnerabilities by type and year [22] . . . . .	13
10	Customized dataset is a subset of all CVEs in NVD . . . . .	14
11	Approach to identify relevant vulnerabilities . . . . .	15
12	Evaluation of HuggingFace sentence-transformers models [28] . . . . .	16
13	Keyword extraction via KeyBERT [25] . . . . .	17
14	Example keyword list . . . . .	18
15	Data entry visualized with JSON Crack . . . . .	19
16	Overview of threat modeling query by using the developed LLM-based model in this study . . . . .	20

## LIST OF TABLES

1	Comparison of current common threat modeling tools [3] . . . . .	3
2	Llama 2 model family details [12] . . . . .	8
3	Example removed stopwords . . . . .	17
4	Dataset size by document for Task 1 . . . . .	21
5	Example questions for Task 1 using custom vulnerability dataset .	22
6	Example questions for Task 2 using reference documents . . . . .	22
7	Example LLM response with and without RAG . . . . .	24
7	Example LLM response with and without RAG . . . . .	25
7	Example LLM response with and without RAG . . . . .	26
7	Example LLM response with and without RAG . . . . .	27
7	Example LLM response with and without RAG . . . . .	28
7	Example LLM response with and without RAG . . . . .	29
7	Example LLM response with and without RAG . . . . .	30
7	Example LLM response with and without RAG . . . . .	31
7	Example LLM response with and without RAG . . . . .	32
7	Example LLM response with and without RAG . . . . .	33
7	Example LLM response with and without RAG . . . . .	34
7	Example LLM response with and without RAG . . . . .	35
8	Example responses for Task 2 . . . . .	38
8	Example responses for Task 2 . . . . .	39
8	Example responses for Task 2 . . . . .	40
9	Example response for Task 2 when insufficient information provided	44

## ACKNOWLEDGMENTS

Thank you to my incredible advisor and mentor, Dr. Zhen Zeng. I'm grateful that you chose to join UW-Milwaukee, and really excited to continue working together in the future.

A huge thanks to my family as well for their support and continued (gentle) nudges that pushed me to finish this.

## 1 Introduction

Threat modeling is an essential part of ensuring products are designed securely. The OWASP Foundation, an initiative 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 and mitigations within the context of protecting something of value [1]. Much of existing security research focuses on risk analysis and prioritization after development or setup, which is essential but results in rework to account for vulnerabilities or threats that should have been identified during design. When threat modeling is done effectively, it should follow the "shift-left security" mentality geared toward identifying and addressing vulnerabilities and threats as early in the product development life cycle as possible.

Since threat modeling can be considered a slow or painful process, often involving digging through multiple types of documents for information, this paper aims to explore whether the process can be made more accessible & more efficient by using Large Language Models (LLM). The target users would be development and security teams conducting threat modeling.

Transportation, specifically the area of connected and autonomous vehicles (CAVs), was selected as the focus because it is becoming more widely researched as vehicle manufacturers compete to extend product capabilities and meet customer expectations. Narrowing in on a specific industry could also lead to more relevant results than training on a broader range of vulnerabilities.

## 2 Background

### 2.1 Threat Modeling Process

CrowdStrike, a well-known cybersecurity technology company that provides threat intelligence and cyberattack response services, defines a threat model as used to profile probable attackers and hackers and to identify both the most likely avenues of attack and the hardware and software most likely to be targeted [2]. They also state the goals of threat modeling, which are below.

- Identify security requirements and vulnerabilities
- Quantify the criticality of threats and vulnerabilities
- Prioritize risk and remediation

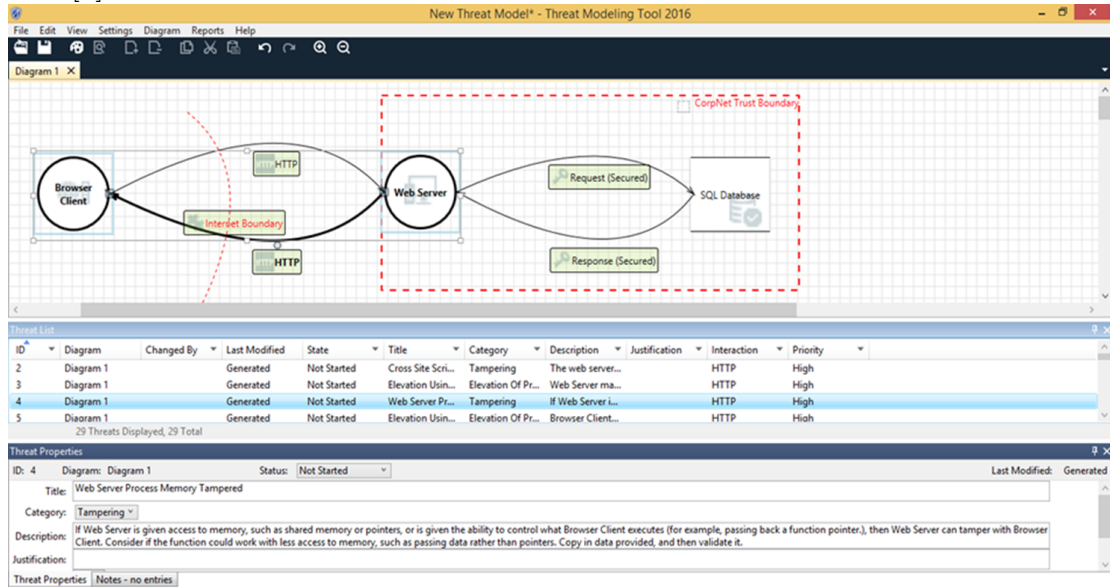
There is existing research comparing current tools that aid in threat modeling, and the criteria they are evaluated against are shown in Table 1 [3].

The Microsoft Threat Modeling Tool (TMT) [5] is one of the oldest tools, initially released in 2014, while many others were created more recently in 2019 and 2020. Some of these 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. An example of this can be seen in Figure 1. 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 & 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) [6].

Table 1: Comparison of current common threat modeling tools [3]

Criterion		TMT	OWASP TD	OVVL	OWASP pytm	Threagile	IriusRisk community edition
Availability — —	Usage	Desktop app	Desktop/web app	Web app	CLI	CLI/server	Web app
	Cross platform	X	✓	✓	✓	✓	✓
	Open source	X	✓	✓	✓	✓	X
Model form		Diagram	Diagram	Diagram	Text based (Python)	Text based (YAML)	Diagram
Model reusability —	Nested components	X	X	X	X	X	✓
	Module import/export	X	X	X	X	X	✓
Threat library —	Source	Self-defined (STRIDE)	Self-defined (STRIDE, LINDDUN, and CIA)	Self-defined (STRIDE) and CVE	CAPEC, CVE, and CWE	CWE	CAPEC, CVE, CWE, and self-defined
	Size	~50	No specific threats	~30 self-defined and CVE list	~100	~40	> 200
Threat evaluation		Manual or predefined	manual	Manual or predefined	Predefined	Predefined and calculation based	Provided, but the mechanism is unclear
Mitigation suggestion		Library dependent*	X	X	✓	✓	✓
Development stage —	Initial release	2014 (TMT 2014)	February 2020 (v1.0)	July 2019 (v1.0)	December 2019 (v1.0)	August 2020 (v1.0)	2015 (v1.0)
	Recent release	July 2020 (v7.3)	August 2021 (v1.5)	July 2019 (v1.0)	April 2021 (v1.2)	August 2020 (v1.0)	September 2021 (v3.14)
SDLC integration		X	GitHub and issue trackers	Issue trackers and vulnerability scanners	X	GitHub and integrated development environment editing support	Issue trackers and testing frameworks

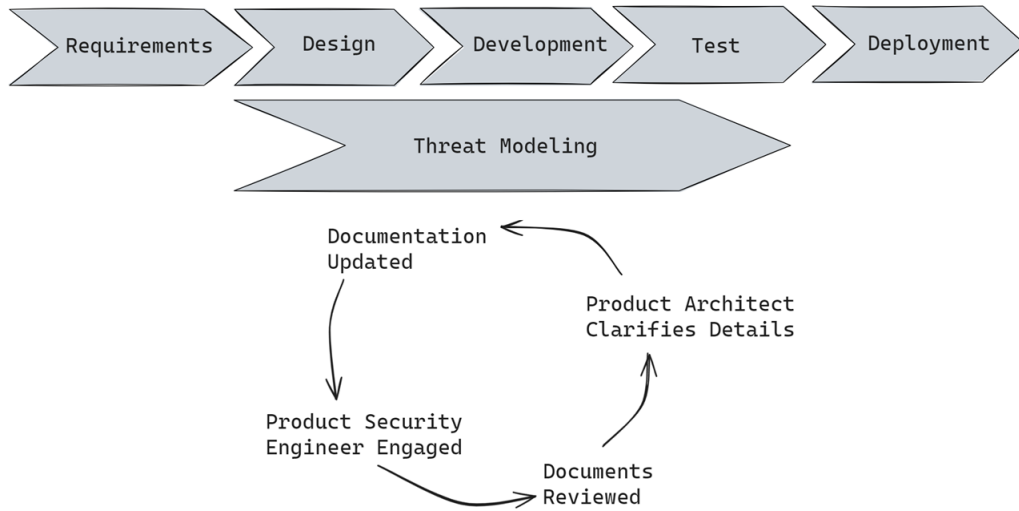
Figure 1: Example threat modeling diagram in the Microsoft Threat Modeling tool [4]



There do not seem to be any tools that try to identify threats based on the provided product design/architecture documentation and scan results, which is part of what this thesis will attempt to explore. This type of documentation can be unstructured and spread across multiple documents, which can make searching through a time-consuming process. Current advances in Large Language Models and other areas within Natural Language Processing may be able to augment the threat modeling process and use of these tools by pulling data from existing documentation instead of requiring users to start from scratch.

In industry, threat modeling typically starts in design and continues through test. The product team starts out with documentation as references 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.

Figure 2: Threat modeling as part of development process

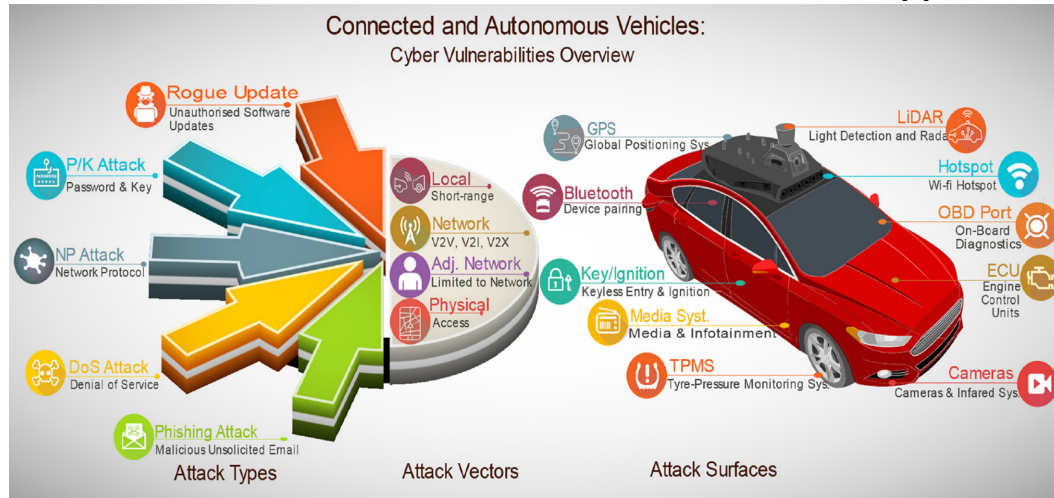


## 2.2 Connected Vehicle Vulnerabilities

Vehicles are becoming more and more complex as consumers seek out ways to connect peripherals, and the possibility of autonomous control becomes a reality. That complexity means larger onboard software and more potential for vulnerabilities. For comparison, while the Windows Vista operating system has only 40 million lines of code and 905 known vulnerabilities in the National Vulnerability Database (NVD) [9], a modern CAV can have 100 million lines of code [8],

Figure 3 is from a research paper that proposes a cyber risk classification framework for connected and autonomous vehicles (CAVs), an overview of CAV attack types, vectors, and surfaces is provided [8].

Figure 3: Overview of connected vehicle vulnerabilities [8]



Many CAVs are also electric cars that rely on charging stations. Recent research conducted by Argonne, Idaho, and Sandia National Laboratories have found these systems vulnerable to malicious firmware updates and that local access could allow hackers to jump from one charger to the whole charger network through the cloud [10].

### 3 Methodology

At the start of threat modeling, development teams can typically share a large amount of documentation as reference for what is being done. The type of threat model created is usually similar across tools because most tools use one of the standard methodologies for threat modeling. The STRIDE method is the most common approach to creating threat models and focuses on what could go wrong [7]. It stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. This methodology shapes the types of questions asked during threat modeling, and one of the main benefits is that it

can be used across industry sectors. Because it involves defining each component of a system and the connections between them, it can be time-consuming to track that down for a complex system. One of the tasks selected for the RAG LLM is to be able to identify components in a system based on provided documentation which could help speed up the process. The vulnerability descriptions in the custom data set generated from the NVD can also be used as a reference for what can go wrong.

To summarize, the proposed method contains two tasks, which have been identified as potentially able to make the threat modeling process easier using natural language processing:

- Task 1: Construct a customized vulnerability data set based on product design documents
- Task 2: Answer queries using multiple reference documents

### **3.1 Model**

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 to be on par with closed-source models when they performed human evaluations [11]. Meta has also been transparent on which external benchmarks were used for evaluation and comparison between their models. Another critical difference from other models is Meta’s emphasis on safety - they provide details on how safety was prioritized during pretraining 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.

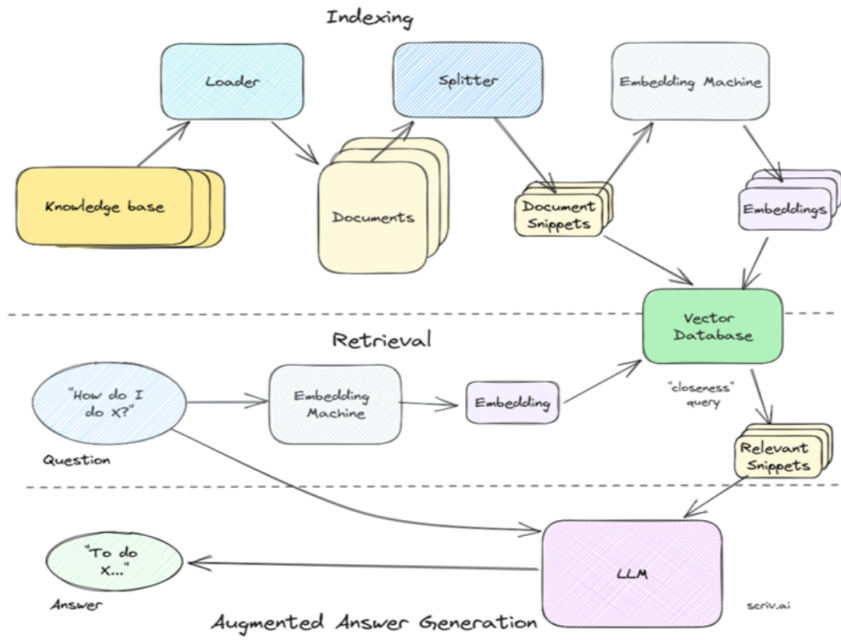
For this research, the pre-trained 7B & 13B Llama 2-CHAT models, fine-tuned versions of Llama 2 that are optimized for dialogue use cases, were run in Google Colab using T4 GPU.

Table 2: Llama 2 model family details [12]

MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	<b>Model architecture:</b>	<b>Data collection for helpfulness and safety:</b>
13B	<b>Pretraining Tokens:</b> 2 Trillion	<b>Supervised fine-tuning:</b> Over 100,000
70B	<b>Context Length:</b> 4096	<b>Human Preferences:</b> Over 1,000,000

The retrieval-augmented generation (RAG) AI framework was used to attempt to give the most accurate and up-to-date information [13]. The RAG model utilizes a pre-trained language model like Llama 2 and integrates it with a retriever component or model, allowing it to access external knowledge sources for context adaption [14].

Figure 4: RAG pipeline [15]



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 [16]. 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.

Figure 5: Common pipeline for a vector database [15]

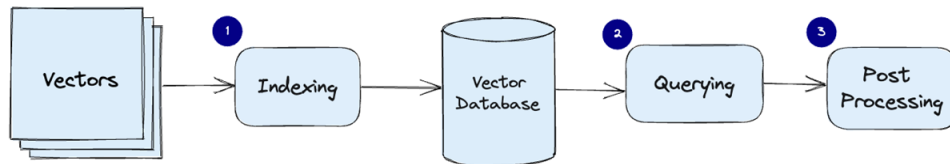
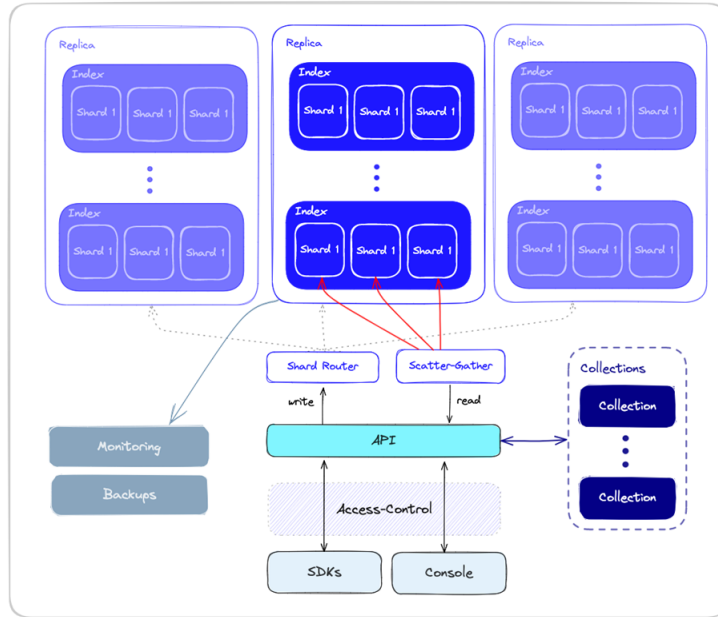


Figure 6 shows the vector database Pinecone, which was used with multiple indexes.

Figure 6: Pinecone architecture [17]



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 this research, the Python library with the functionality below was used.

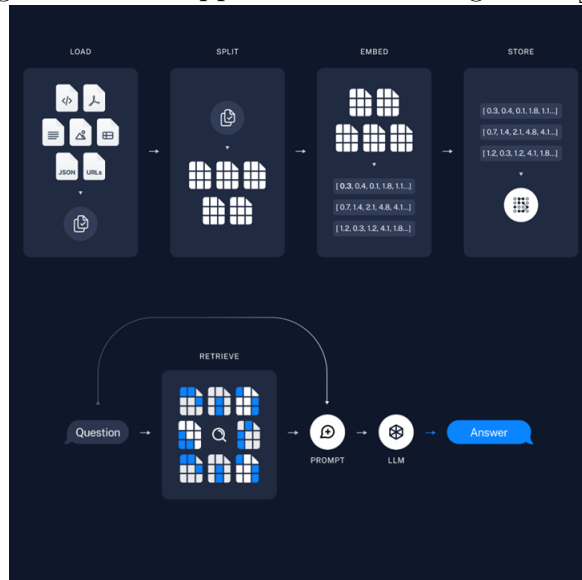
- Document loading: load unstructured data from sources like PDFs
- Text splitting: chunk data by size or characters as part of document transformers
- Vector store-backed retriever: read from a vector store like Pinecone

The specific loader used in this research is the PyPDFDirectoryLoader since the intention is to be able to load multiple related documents at once for context. 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' with 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 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 [19]. Using the retriever langchain.vectorstores, Pinecone was specified as the vector store type. The index\_name, embeddings, and field in metadata for the text chunks that contained text content were passed in.

Figure 7: RAG application with LangChain [18]

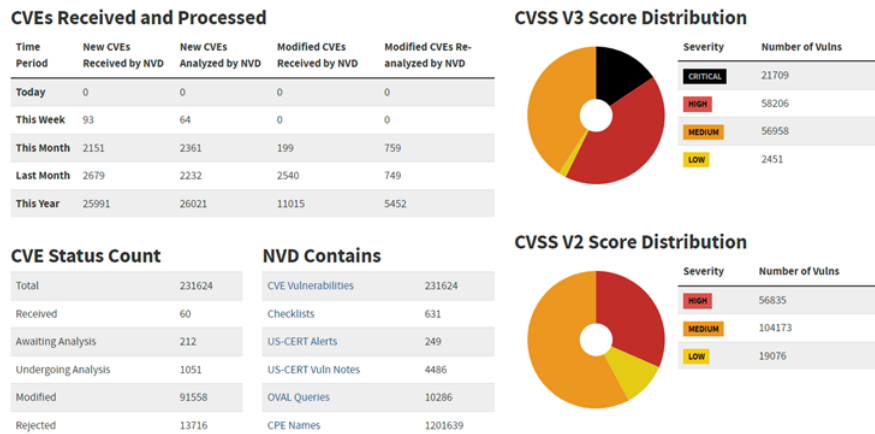


### 3.2 Data

The NVD is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol

(SCAP). This data enables automation of vulnerability management, security measurement, and compliance [20]. There are over 231,000 vulnerabilities in the NVD. Each vulnerability is associated with a Common Vulnerabilities and Exposures (CVE) identifier, a unique ID. For each CVE, there are 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.

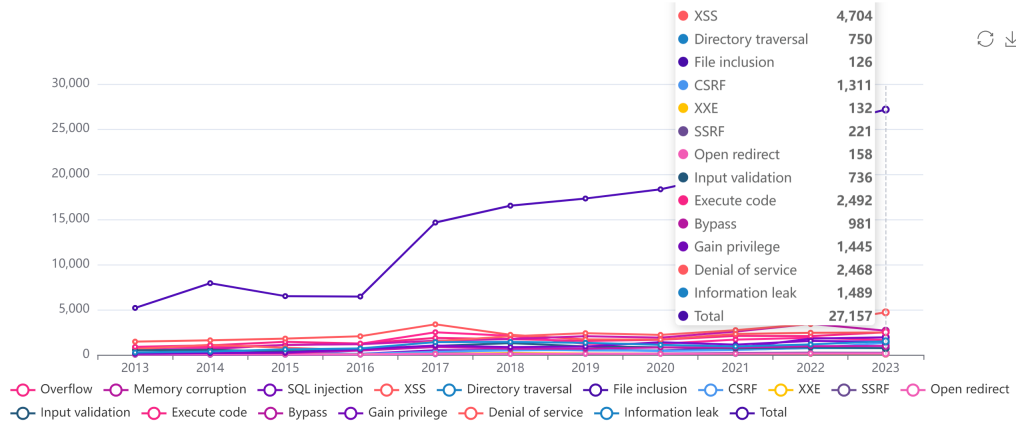
Figure 8: NVD dashboard [21]



The NVD is not a static database; it is continuously updated and dashboards have been created to quickly visualize and track vulnerabilities as they appear each year. At the time of this writing, in December 2023, 27,000 vulnerabilities were added throughout the year. In Figure 9, it can be seen that the number of vulnerabilities tracked in the NVD is increasing each year. Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website and is the vulnerability type that

increased the most over the last year [22].

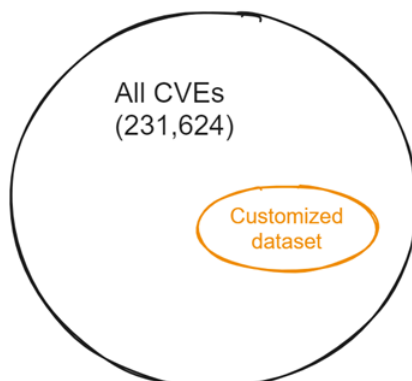
Figure 9: NVD vulnerabilities by type and year [22]



Because the NVD is central to most research related to security vulnerabilities, threats and risk, there are multiple studies that attempt to identify patterns and evolving trends using the NVD. One study did an exploratory data analysis of the vulnerabilities identified in the first six months of 2022 with the goal being to identify patterns that would help in preparing the data for future machine learning use cases [23]. Another study focused on identifying emerging trends by proposing an integrated data mining framework to automatically lay out how vulnerabilities develop over time [24].

A subset of what is available in the NVD was used to create the customized dataset available to the LLM. The subset is based on searching the NVD with keywords extracted from the provided documentation.

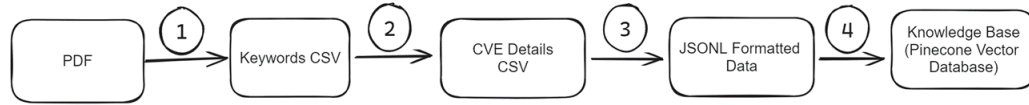
Figure 10: Customized dataset is a subset of all CVEs in NVD



To construct the dataset used for Task 1, scripts ( 700 lines) were developed that go through the steps below to construct a customized vulnerability dataset as input for the RAG LLM. 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 main workflow of the developed script is shown as follows.

- Step 1: Keywords extracted from documentation
- Step 2: NVD API programmatically queried with the top 15 extracted keywords; Relevant fields extracted from response to generate a CSV
- Step 3: Data preprocessed to get in the required format for the model
- Step 4: JSONL file specified as input to be accessible to the LLM via Pinecone vector database

Figure 11: Approach to identify relevant vulnerabilities



Several NLP techniques shown as follows have been utilized to extract keywords from the input document.

- KeyBERT library with the all-mpnet-base-v2 model
- Remove English stopwords
- Remove “Expert defined”/custom stopwords
- Stemming

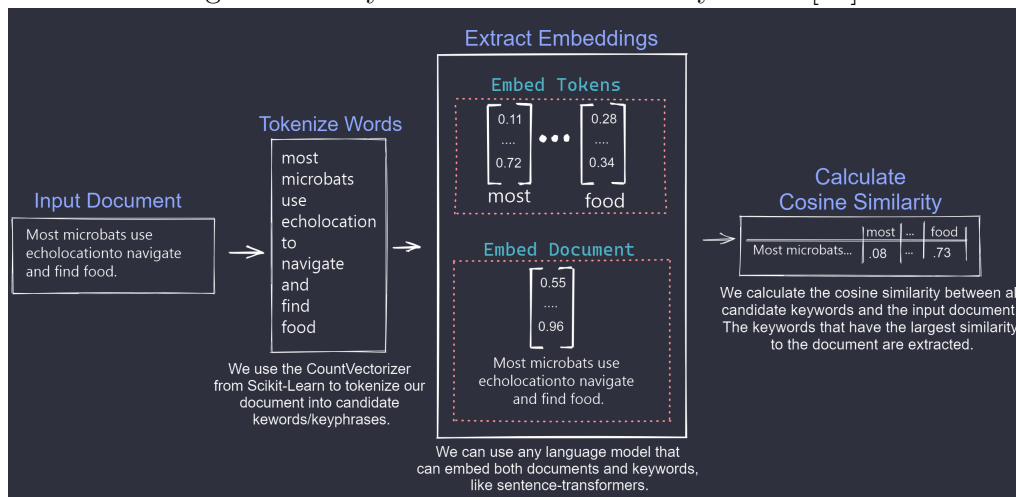
KeyBERT depends on Bidirectional Encoder Representations from Transformers (BERT)-embeddings and cosine similarity to find the keywords and key phrases in a document that are most similar to the document itself [25]. Embeddings are numerical representations of pieces of information, such as text or documents, that capture semantic meaning. The all-mpnet-base-v2 model was selected when using the KeyBERT library based on the performance, speed and model size comparison seen in Figure 12.

Figure 12: Evaluation of HuggingFace sentence-transformers models [28]

Model Name	Performance Sentence Embeddings (14 Datasets) <sup>1</sup>	Performance Semantic Search (6 Datasets) <sup>1</sup>	📊 Avg. Performance <sup>1</sup>	Speed <sup>1</sup>	Model Size <sup>1</sup>
all-mpnet-base-v2 <sup>1</sup>	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1 <sup>1</sup>	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1 <sup>1</sup>	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2 <sup>1</sup>	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1 <sup>1</sup>	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2 <sup>1</sup>	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1 <sup>1</sup>	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2 <sup>1</sup>	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2 <sup>1</sup>	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2 <sup>1</sup>	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2 <sup>1</sup>	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1 <sup>1</sup>	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2 <sup>1</sup>	60.18	27.35	43.77	4000	480 MB

KeyBERT has been widely used in academic research, with examples being in studies for security bug reports prediction and automatic domain-specific corpora generation from Wikipedia [29] [30]. Using KeyBERT, these studies were able to extract relevant keywords that captured semantic information from the input text efficiently. Because of its gaining popularity, a recent study systematically tested the different embedding models for keyword extraction using KeyBERT [31]. It does not compare the specific models used for this research, but its contributions cover introducing an extensive comparative study on embedding models supported by KeyBERT and observing the text length effect on the accuracy of keyword extraction, both of which could be useful to apply to the selected models in this research as future work.

Figure 13: Keyword extraction via KeyBERT [25]



When using KeyBERT, two types of stopwords are removed: "English" stopwords as defined by the NLTK (Natural Language Toolkit) and "expert-defined" stopwords, a list created during this research via manual inspection of keyword lists generated. NLTK supports removing stopwords in other languages, but since only English vulnerability descriptions were analyzed, only the "english" stopwords corpus was used. These two stopwords lists were combined and set as a parameter for KeyBERT. In the "expert-defined" stopword list, the words tended to be security-related, but a few, like "allow" and "architecture," were more general. Examples of these stopwords can be found in Table 3.

Table 3: Example removed stopwords

Stopword Type	Example
NLTK "english"	['start', 'but', 'to', 'and', 'the', 'in', 'it', 'are', 'I', 'that']
Custom "expert-defined"	['allow', 'attack', 'hacker', 'security', 'cyberattack', 'cyber', 'cybersecurity', 'intelligence', 'architecture', 'secure']

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 would allow for key phrases, which could help more with context. This initial research tried to maximize a subset of vulnerabilities, so keywords were selected over key phrases. An example of keywords extracted from a PDF and the calculated similarity score is below as shown in Figure 14.

Figure 14: Example keyword list

```
1 keyword,score
2 vehicle,0.3199
3 sensor,0.3075
4 driver,0.3028
5 svm,0.3016
6 kalman,0.2995
7 brake,0.2811
8 autonomous,0.2794
9 ddos,0.2646
10 iot,0.26
11 lidar,0.2598
12 roadway,0.2379
13 traffic,0.2377
14 passenger,0.2304
15 van,0.2304
16 gps,0.2251
```

Stemming via NLTK was also an important step to avoid running keyword searches that returned duplicate results. An example of stemming would be only including the keyword "vehicle" when both "vehicle" and "vehicles" appear. A slightly more complicated example would be only including "drive" when "driving", "drives," and "drivers" all appear.

The model input is formatted as a JSONL (JSON Lines) file, which will be read by the LLM for further operation. Each entry in the JSONL file corresponds to a 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 [26]. The JSONL file has an example entry below as shown in Figure 15.

Figure 15: Data entry visualized with JSON Crack

```
pdf.keyword: "vehicle"
cve.id: "CVE-2018-18203"
cve.published: "2018-11-28T23:29:00.357"
cve.descriptions: "A vulnerability in the update mechanism of Subaru StarLink Harman head u...
cve.descriptions.keywords: "['vulnerability', 'update', 'mechanism', 'subaru', 'starlink', ...
cve.metrics.cvssMetricV2.vectorString: "AV:L/AC:M/Au:N/C:C/I:C/A:C"
cve.metrics.cvssMetricV2.accessVector: "LOCAL"
cve.metrics.cvssMetricV2.accessComplexity: "MEDIUM"
cve.metrics.cvssMetricV2.baseScore: "6.9"
cve.metrics.cvssMetricV2.exploitabilityScore: "3.4"
cve.metrics.cvssMetricV30.vectorString: "CVSS:3.0/AV:P/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H"
cve.metrics.cvssMetricV30.attackVector: "PHYSICAL"
cve.metrics.cvssMetricV30.attackComplexity: "HIGH"
cve.metrics.cvssMetricV30.baseScore: "6.4"
cve.metrics.cvssMetricV30.exploitabilityScore: "0.5"
cve.metrics.cvssMetricV30.impactScore: "5.9"
cve.references: [{"url": "https://github.com/sgayou/subaru_starlink_research"}, {"sourc...
chunk-id: 7933
```

Each PDF would be read, and the keywords extracted using the keyBERT library. English stopwords and custom security stopwords were removed. The remaining keywords, as well as "expert defined" keywords, were then used to search the NVD via API and retrieve CVEs with their descriptions. That data was then further processed to be in the required format expected by the model.

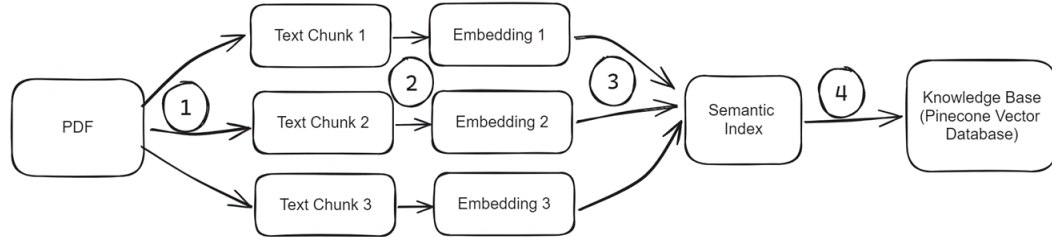
The goal of the second task was to answer the query questions of threat modeling by using our developed LLM-based model. The steps below were used

- Load and extract text from PDF. Split and chunk text by size.
- Create embeddings (vectors used to compress size) for each text chunk
- Create Pinecone index; an index is the highest-level organizational unit of

vector data in Pinecone [27]

- Push data to Pinecone vector database to make it accessible by LLM

Figure 16: Overview of threat modeling query by using the developed LLM-based model in this study



## 4 Results

For initial testing, 10 PDF documents were used, and 3 questions were asked of each task-related model for a total of 60 queries. The PDFs ranged from three to seven pages, with an average of five pages. All of the documents were in English and included figures, and most included tables. The documents were sourced from IEEE Xplore and Cybersecurity & Infrastructure Security Agency site [32] [33].

The PDFs were treated as independent of one another because of the availability of related documentation. If this process were to be used for threat modeling, the goal would be to include as much documentation for a product as possible for each task. When documents are loaded, the LangChain loader points to a directory, so any PDF included in that directory will be loaded, and the metadata will tie the extracted text to that document and page number.

Table 5 and Table 6 shows the example of questions for both Task 1 and Task 2. These questions cover different cybersecurity areas, including secrets management, identity & access control, network, and working scenarios. Specifically for the

system design and architectures, questions in Task 2 contain the area of system description, system environment, and data. For Task 1, the number of entries in the custom vulnerability dataset ranged from 1,000 to 36,000, depending on the PDF as shown in Table 4. The total number of CVEs in the NVD is 231,000.

Table 4: Dataset size by document for Task 1

Document	Number of Vulnerabilities
1	2,260
2	14,060
3	35,924
4	7,773
5	29,970
6	10,077
7	1,022
8	1,336
9	2,958
10	4,049

If the "top\_n" parameter passed to KeyBERT is increased, then there is an increase in returned keywords which means an increase in results from the NVD based on those additional searches. The issue with that is increasing that number could be less relevant results.

Table 5: Example questions for Task 1 using custom vulnerability dataset

Model	Area	Questions
Task 1	Secrets management	‘How can shared private keys used for authentication be exposed?’
Task 1	Identity & access control	‘What are some concerns related to open source authentication methods?’
Task 1	Network	‘Other than a hash function and timestamp, what can be used to avoid a replay attack?’
Task 1	Network	‘What vulnerabilities exist related to Modbus/TCP?’
Task 1	Network	‘What types of attacks are most common when targeting Modbus communication and what best practices could help minimize risk?’
Task 1	Scenario	‘What types of vulnerabilities are associated with vehicle to infrastructure communication?’

Table 6: Example questions for Task 2 using reference documents

Model	Area	Questions
Task 2	System description	‘Summarize what is being done’
Task 2	System description	‘What components are involved in the design?’
Task 2	System environment	‘How is the encrypted connection established?’
Task 2	System description	‘What concerns are there based on the attack surface?’
Task 2	System description	‘How is the the custom Modbus TCP/IP protocol stack implemented?’
Task 2	Data	‘How is data secured during transmission?’

#### 4.1 Evaluation

Human evaluation was usually adopted for evaluating large language models (LLMs). Through human evaluation, various aspects of the model’s responses, such as accuracy, coherence, relevance, and fluency are assessed by individuals

independently. The method of human evaluation is essential in determining how well the model can meet user expectations and needs. The output of human evaluation can provide feedback for further development and refinement of LLM. In this study, human evaluation was adopted to analyze these initial results.

To evaluate the performance of the proposed framework of LLM-based threat modeling in Task 1, results are compared between the base LLM with and without Retrieval Augmented Generation (RAG). Due to the page limits, three questions & answers are coded and compared in Table 7. The responses shown in Table 7 where the "Document-Question Pair" column is "D1-Q1" were returned for question Q1, 'Other than a hash function and timestamp, what can be used to avoid a replay attack'. This question aims to check that the selected mitigation strategy actually addresses the threat of a replay attack and to provide additional options that may not have been considered. This 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. Regarding the input document associated with this question, the keywords from each PDF were extracted. During the evaluation, the input document has been manually read.

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q1. Other than a hash function and timestamp, what can be used to avoid a replay attack?	Without RAG	?/n/nI'm looking for a way to prevent replay attacks in my system. I have already implemented a hash function and a timestamp to ensure the message is not tampered with and is fresh. However, I want to know if ... mind:/n/n1. Nonce: Use a nonce (number used once) in the message. This would ensure that the message can only be processed once, ... /n8. Secure channel: Establish a secure communication channel, such as SSL/TLS, to protect the message from interception and tampering./n/nAre there any other measures that can be taken to prevent replay attacks? Are there any drawbacks or limitations to the above solutions?/n/nAnswer:/n/nReplay attacks can be prevented using various techniques, including those ...	using nonce, sequence number, random token, digital signature, encryption, integrity check ...

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q1. Other than a hash function and timestamp, what can be used to avoid a replay attack?	With RAG	A challenge-response system can be used to avoid a replay attack. In such a system, the server sends a random challenge to the client along with each request, and the client must respond with a valid response before the request is processed. This makes it difficult for an attacker to replay a previous request, as they would need to know the challenge and response used in the original request. Additionally, using a sliding window of challenges can help prevent replay attacks over a long period of time.	challenge-response system. Server sends random challenge.

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q2. How can shared private keys used for authentication be exposed?	Without RAG	to the public?/n/nI'm studying for a security certification and I came across this question:/n/n"How can shared private keys used for authentication be exposed to the public?" /n/nI understand that shared private keys are not supposed to be exposed to the public, as they are sensitive information that should be kept confidential. However, I'm having trouble ... /n5. Insider threats: An authorized user with access to the shared private key may intentionally or ... /n5. Legal liability: Depending on the specific laws and regulations applicable to the organization, exposure of shared private keys may lead to legal liability and financial penalties./n/nTo mitigate these risks, organizations	unsecured storage, network breaches, malware attacks, social engineering, insider threats

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q2. How can shared private keys used for authentication be exposed?	With RAG	Shared private keys used for authentication can be exposed through various means, including man-in-the-middle attacks, spoofing vulnerabilities, and unintentional sharing of screen contents. Additionally, unsecured storage of private keys and passwords can also lead to exposure. It is important to implement strong security measures, such as encryption and access controls, to protect against these types of threats.	man-in-the-middle attack, spoofing vulnerabilities, unintentional sharing of screen contents, unsecured storage

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q3. What are some concerns related to open source authentication methods?	Without RAG	?/n/nOpen-source authentication methods, such as OAuth and OpenID Connect, have gained popularity in recent years due to their flexibility and ease of use. However, there are also some concerns related to their adoption:/n/n1. Security risks: As with any software, there is a risk of security vulnerabilities in the code that could be exploited by attackers. This is .../n10. Lack of transparency: It can be difficult to understand how open-source authentication methods handle user data and authentication... ensure successful implementation and ongoing maintenance.	security risks, lack of support, compatibility issues, limited scalability, ... long-term maintenance

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D1-Q3. What are some concerns related to open source authentication methods?	With RAG	Based on the provided information, there are several concerns related to open source authentication methods, including:  Vulnerabilities in specific versions of Mahara, One-Time Passwords In Everything (OPIE), and mod_auth_shadow that can be exploited by remote attackers to gain unauthorized access to systems or services.  The potential for attackers to determine the existence of user accounts using OPIE and the possibility of bypassing security restrictions using mod_auth_shadow.  The need for proper restriction of excessive authentication attempts to prevent legitimate users from being locked out of systems or services due to continuous unauthorized login attempts.	security risks, lack of support, compatibility issues, limited scalability, ... long-term maintenance

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q1. What are some concerns related to open source authentication methods?	Without RAG	communication? Modbus is a widely used protocol for remote monitoring and control of industrial equipment. Modbus/TCP is an extension of the Modbus protocol that uses TCP/IP for communication over Ethernet networks. While Modbus/TCP offers many benefits, it also presents some security risks. Here are some common vulnerabilities related to Modbus/TCP communication: 1. Unauthorized Access: Modbus/TCP devices often use default or weak passwords, which can be easily guessed by attackers. This allows unauthorized access to the device and potentially sensitive information. 2. Buffer Overflow Attacks: ... 3. Integrity Checks: Modbus/TCP devices may not perform integrity	unauth- orized access, buffer over- flow attacks, com- mand injec- tion, ... , out- dated firmware, weak pass- ward recov- ery ...

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q1. What are some concerns related to open source authentication methods?	With RAG	<p>/n/nThere are several vulnerabilities related to Modbus/TCP that have been identified and documented. These include:/n/n* CWE-754: Improper Check for Unusual or Exceptional Conditions, which could cause denial of service of the controller when communicating over the Modbus TCP protocol./n* CWE-191: Integer Underflow (Wrap or Wraparound), which could cause a denial of service of the controller due ... consult the relevant documentation and security advisories for more information on these vulnerabilities and how to address them.</p>	CWE-754, CWE-191, unauthorized user sending malformed message,

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q2. What concerns are there when using OpenPLC and ScadaBR?	Without RAG	OpenPLC and ScadaBR are open-source software solutions for building automation and control systems. While they offer many benefits, there are also some concerns that should be considered when using these tools: 1. Security: As with any networked system, security is a major concern when using OpenPLC and ScadaBR. The software must be properly configured and ... 9. Integration with existing systems: If you already have an existing building automation system, integrating OpenPLC or ScadaBR with it may be challenging. It's important to evaluate the potential impact on your existing system before making any changes.	security, inter-operability, compatibility, scalability, training and support, customization ...

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q2. What concerns are there when using OpenPLC and ScadaBR?	With RAG	There are several concerns when using OpenPLC and ScadaBR. Firstly, there is a buffer overflow vulnerability in the OpenPLC controller that can cause a runtime crash of the PLC or have unspecified other impact. Additionally, there is a vulnerability in SIMATIC ET 200SP Open Controller ... resulting in a denial-of-service condition on the network stack. These vulnerabilities could be exploited by an attacker with no privileges and no user interaction, potentially leading to a compromise of the network connectivity.	buffer overflow vulnerability, runtime crash, SIMATIC ET 200SP Open Controller ... denial-of-service (DoS)

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q3. What types of attacks are most common when targeting Modbus communication and what best practices could help minimize risk?	Without RAG	?/nModbus is a widely used communication protocol in industrial control systems (ICS) that operates over RS-232 or RS-485 serial connections. It is commonly used for remote monitoring and control of equipment .../nTypes of Attacks on Modbus Communication:/n1. Eavesdropping: An attacker can intercept and monitor Modbus traffic to gain unauthorized ... /n6. Malware: An attacker can install malware on the Modbus device to steal sensitive information, disrupt operations, or gain unauthorized access to the system./nBest Practices to Minimize Risks:/n1. Secure Communication Channels: ... /n4. Firewalls and Access Control Lists (ACLs): Implement firewalls and ACLs to restrict access ... damage that an attacker	attacks on modbus communication: eavesdropping, replay attacks, man-in-the-middle attacks, denial of service (DoS) attacks, ...

Table 7: Example LLM response with and without RAG

Document- Question Pair	Response Type	Response	Codes
D2-Q3. What types of attacks are most common when targeting Modbus communication and what best practices could help minimize risk?	With RAG	Based on the given text, the most common attacks targeting Modbus communication include spoofing or replay attacks, brute-force attacks, and DoS attacks. To minimize risk, best practices such as validating devices before establishing Modbus communications, using secure protocols and encryption, and implementing access controls and firewalls can help protect against these types of attacks	attacks on modbus com- muni- cation: spoof- ing or replay attacks ... De- fense: validat- ing ...

For D1-Q1, acceptable answers could include the use of encryption, sliding window, or Nonce, so both responses without RAG and with RAG contained correct answers. Part of the answer in the response without RAG was not entirely correct in that digital signatures, by themselves, cannot prevent a replay attack because a signed message can be captured and resent [34]

For D1-Q2, acceptable answers could include unsecured storage, which is mentioned in both responses. The response without RAG also includes details on the potential consequences of exposing shared private keys, which isn't directly part of the question and not included in the RAG response.

For D1-Q3, acceptable answers could be general concerns related to the use of open-source software, such as lack of support, lack of thorough testing/review, and limited scalability. The response without RAG covers this, but the response with RAG seemed to be referencing something that was not relevant. It's true that proper restriction of excessive authentication attempts is a good practice, but Mahara, and `mod_auth_shadow` are not relevant for this question. To try to understand why those ended up in the response, the `vectorstore.similarity_search` was examined, which would give a good general idea for what documents were being referenced for this question. The italicized text below is the output of `vectorstore.similarity_search` for the question "What are some concerns related to open source authentication methods":

*[Document(page\_content="Mahara 15.04 before 15.04.8 and 15.10 before 15.10.4 and 16.04 before 16.04.2 are vulnerable to some authentication methods which do not use Mahara's built-in login form still allowing users to log in even if their institution was expired or suspended.", metadata='pdf.keyword': 'ev'), Document(page\_content='One-Time Passwords In Everything (a.k.a OPIE) 2.32 and 2.4 allows remote attackers to determine the existence of user accounts by printing random passphrases if the user account does not exist and static passphrases if the user account does exist.', metadata='pdf.keyword': 'ev'), Document(page\_content='The mod\_auth\_shadow module 1.0 through 1.5 and 2.0 for Apache with AuthShadow enabled uses shadow authentication for all locations that use the require group di-*

*rective even when other authentication mechanisms are specified which might allow remote authenticated users to bypass security restrictions.', metadata='pdf.keyword': 'ev']]*

From this output, it can be seen that the vulnerabilities referenced are based on the 'ev' keyword. This keyword is actually meant to refer to Electric Vehicles (EV) instead of other words that contain the pattern 'ev' such as 'even' and 'everything'. The NVD API keyword search is case-insensitive, so excluding acronyms in the future may produce more accurate results.

Task 2, focused on querying multiple documents for information, was evaluated by reviewing the accuracy of the responses. The values in Table 8 are example responses that include the output of `vectorstore.similarity_search` to show what portions of the document are being referenced. Running a `similarity_search` with the question helps avoid evaluating the results in a black box, instead highlighting the portion of the document that is being referenced.

Table 8: Example responses for Task 2

Document- Question Pair	Vectorstore similarity_search	Response	Accurate
<p>D1-Q1. How is the encrypted connection established?</p>	<p>[Document(page_content="trust. The methods for establishing an encrypted connection between an Aggregator and an EV are provided below. /n1. A message seeking electricity from EVs is signed using /nan Aggregator's private key and its certificate. /n2. A neighboring EV receives the message and verifies ... /nthanks to the project's hardware and software design: User /nsecure identification; Temperature; Battery state-of-charge;")]</p>	<p>Based on the provided context, the encrypted connection between an Aggregator and an EV is established using the following steps:/n/n1. The Aggregator signs a message .../n3. When an EV decides to sell electricity to an Aggregator, it uses the Diffie-Hellman key exchange to establish ... /nTherefore, 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.</p>	<p>Yes</p>

Table 8: Example responses for Task 2

Document- Question Pair	Vectorstore similarity_search	Response	Accurate
<p>D1-Q2. How are re- play attacks avoided?</p>	<p>[Document(page_content ='shared key using the Diffie-Hellman method, and then /encrypts and decrypts data using the keyed, symmetric block /cypher Blowfish. ... A hash function and a timestamp /are used to avoid a replay attack on the EV ... /It handles new and unique issues, such as latency, which is /criti- cal for secured wireless communications with fast-moving')]</p>	<p>To avoid replay attacks, the system uses a hash function and a timestamp when the EV transmits the required data to the Power Aggre- gator. The hash function takes the EV's unique identi- fier, the IDE, and the times- tamp as input and produces a fixed-size hash value. This hash value is sent along with the other data to the Power Aggregator... Power Aggre- gator can verify the digi- tal signature using the same shared secret key, and if it matches, it knows that the data has not been tampered with and that the replay at- tack has been avoided.</p>	<p>Yes</p>

Table 8: Example responses for Task 2

Document- Question Pair	Vectorstore similarity_search	Response	Accurate
D1-Q3. How is the secret key protected?	[Document(page_content="method to generate a shared secret that is accessible by /nboth parties. /n4. Now, the Aggregator and EV ... will utilize /ntheir secret key. /n5. The EV transmits the required data to the Power /nAggregator, including the EV's unique identifier, the /nIDE, and a timestamp. A hash function and a timestamp /nare used to avoid a replay attack on the EV... using the Diffie-Hellman method, ...')]	Based on the provided context, the secret key used for encrypting and decrypting messages between the Aggregator and the Electric Vehicle (EV) is protected as follows:/n/n1. The secret key is generated using a hash function, which takes the EV's unique ... The Aggregator and EV use the shared secret key to encrypt and decrypt messages, but they do not store the key in plaintext ... protected using a combination of cryptographic hash functions, one-way functions, and secure storage ...	Yes

## 4.2 Analysis

### 4.2.1 Observations made for Task 1

Task 1 has been evaluated by comparing results from the base LLM with and without Retrieval Augmented Generation (RAG). The result highlights the differences in the output performance and characteristics. The RAG-enhanced LLM produced responses that were not only more concise, averaging 74 words compared to the 253 words for the base LLM, but also more precise, often incorporating specific and relevant details from the custom dataset. The base LLM generates lengthier content with irrelevant or hallucinations.

In this pilot study, with the limited sample size (10 pdf inputs) used, one hallucination was seen for RAG LLM responses. Existing studies with larger datasets have shown RAG LLMs return fewer hallucinations than LLMs without RAG. Moreover, to evaluate hallucinations, besides human evaluation, companies like Vectara are working on ways to evaluate hallucinations in RAG LLMs; Vectara has open-sourced their Hallucination Evaluation Model (HEM) for this use case [36]. These findings suggest that RAG LLM offers a more accurate and contextually relevant approach for threat modeling tasks. The comparison details are summarized as follows.

- RAG responses were more concise and, at times, included descriptions of relevant vulnerabilities from the private database generated using the custom dataset
- RAG responses sometimes lost some of the detail included in LLM without RAG responses

- LLM without RAG tended to be lengthier and included hallucinations; the average RAG response included 74 words vs an average of 253 words for responses not using RAG

#### 4.2.2 Observations made for Task 2

In the evaluation of Task 2, the analysis focused on response accuracy. This study revealed that query response times ranged from 25 to 50 seconds. This variation in response time can be affected by several factors, including the way of storing data and the specific configurations of the Pinecone database for these tasks. Additionally, the network connection between Google CoLab and Pinecone might play a critical role in these observed time frames. Further investigation is needed to identify acceptable response times and understand what is contributing to these longer-than-desired times.

Regarding the accuracy of the model's content, the overall results are positive. When the RAG LLM model was tasked with summarizing the system design and architecture components in a query, it usually returned accurate and relevant responses. Based on this observation in this study, the RAG LLM model shows capability in handling threat modeling-related queries effectively. More interestingly, the RAG LLM model shows the capability of the "awareness of honesty" in instances where questions were not supported by sufficient information in the input documentation. An example of this can be seen in Table 9, as

*Based on the provided context, I don't know the answer to your question. The passage does not provide any information about ...*

In this instance, the model communicated its inability to provide an accurate

response that it did not know and attempted to explain the reason behind this limitation.

Table 9: Example response for Task 2 when insufficient information provided

Document-Question Pair	Vectorstore similarity_search	Response
How is the custom Modbus TCP/IP protocol stack implemented?	[Document(page_content='have now evolved to function using routable protocols that/nuse Ethernet, e.g., TCP/IP and UDP/IP. Modbus TCP is one/nof the oldest industrial network protocols. Many industrial/nprocesses use the Modbus TCP protocol for communication/namong devices ... These advancements, however,/ncome with a price. , ... system testbed to validate critical infrastructure protection')]]	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.

### 4.3 Limitations

Large language models can provide many benefits when it comes to making the process of threat modeling easier, but they also have weaknesses that could affect the quality of their output.

One significant area that is concerning is related to biases that can arise 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 [37]. 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 area of concern is hallucinations, incorrect responses generated by LLMs that can be due to biased datasets, lack of real-world knowledge, or proper understanding of input. Research to decrease the frequency of hallucinated responses is actively being pursued, with one approach being to use guided iterative prompting [38]. There are situations where 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 [39].

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 [40].

A significant barrier to scaling LLMs is the cost and time to train [41]. 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.

## **5 Future Work**

Further explorations can be considered for enhancing the effectiveness and applicability of this RAG LLM solution for threat modeling. The technique challenges are rooted in investigating how to improve the query response accuracy and time. Additionally, future work also needs to enable the interface to handle a variety of tasks and to improve the user experience. Another critical area involves investigating different models for extracting keywords from PDFs for constructing the customer vulnerability database to ensure the most accurate and relevant information is obtained.

## **6 Conclusion**

This study is a pilot study to investigate large language model-assisted threat modeling. Based on our best knowledge, this study is the first solution of threat modeling by leveraging the power of a large language model. This approach extracts the domain knowledge of given PDF documents by constructing a customized vulnerability dataset for each single input. The insights gained from comparing base LLMs with the RAG LLMs, alongside the findings on response times and model accuracy reveal the potential of using LLMs for supporting threat modeling tasks.

## REFERENCES

- [1] Drake, V. (n.d.). Threat modeling. OWASP Foundation. Retrieved October 9, 2023, from [https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)
- [2] “What Is Threat Modeling? - CrowdStrike,” crowdstrike.com. Accessed: Oct. 10, 2023. [Online]. Available: <https://www.crowdstrike.com/cybersecurity-101/threat-modeling/>
- [3] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, “Threat Modeling Tools: A Taxonomy,” *IEEE Secur. Privacy*, vol. 20, no. 4, pp. 29–39, Jul. 2022, doi: 10.1109/MSEC.2021.3125229.
- [4] “Application Threat Modeling,” Optiv. Accessed: Nov. 29, 2023. [Online]. Available: <https://www.optiv.com/insights/discover/blog/application-threat-modeling>
- [5] jegeib. (2022, August 25). Microsoft Threat Modeling Tool overview - Azure. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool>
- [6] L. P. da Silva, B. S. Nascimento, R. A. M. P. Dias, and D. 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)*, Oct. 2022, pp. 01–06. doi: 10.1109/WF-IoT54382.2022.10152291.
- [7] “STRIDE (security),” Wikipedia. Nov. 02, 2023. Accessed: Nov. 10, 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=STRIDE\\_\(security\)&oldid=1183095120](https://en.wikipedia.org/w/index.php?title=STRIDE_(security)&oldid=1183095120)
- [8] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, “Connected and autonomous vehicles: A cyber-risk classification framework,” *Transportation Research Part A: Policy and Practice*, vol. 124, pp. 523–536, Jun. 2019, doi: 10.1016/j.tra.2018.06.033.
- [9] T. Zimmermann, N. Nagappan, and L. Williams, “Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista,” in *2010 Third International Conference on Software Testing, Verification and Validation*, Paris, France: IEEE, 2010, pp. 421–428. doi: 10.1109/ICST.2010.32.

- [10] “Sandia studies vulnerabilities of electric vehicle charging infrastructure.” Accessed: Oct. 12, 2023. [Online]. Available: [https://newsreleases.sandia.gov/ev\\_security/](https://newsreleases.sandia.gov/ev_security/)
- [11] H. Touvron, L. Martin, and K. Stone, “Llama 2: Open Foundation and Fine-Tuned Chat Models”.
- [12] “Llama 2,” Meta AI. Accessed: Nov. 2, 2023. [Online]. Available: <https://ai.meta.com/llama-project>
- [13] “What is retrieval-augmented generation?,” IBM Research Blog. Accessed: Nov. 24, 2023. [Online]. Available: <https://research.ibm.com/blog/retrieval-augmented-generation-RAG>
- [14] “Empowering Domain Adapted LLM’s.” Accessed: Nov. 24, 2023. [Online]. Available: <https://www.arcee.ai/blog/the-pros-and-cons-of-retrieval-augmented-generation-and-fine-tuning-with-llms>
- [15] “How do domain-specific chatbots work? An Overview of Retrieval Augmented Generation (RAG),” Scriv. Accessed: Nov. 26, 2023. [Online]. Available: <https://scriv.ai/guides/retrieval-augmented-generation-overview/>
- [16] “What Is Retrieval-Augmented Generation (RAG)?” Accessed: Nov. 26, 2023. [Online]. Available: <https://www.oracle.com/artificial-intelligence/generative-ai/retrieval-augmented-generation-rag>
- [17] R. Schwaber-Cohen, “What is a Vector Database & How Does it Work? Use Cases + Examples — Pinecone.” Accessed: Nov. 26, 2023. [Online]. Available: <https://www.pinecone.io/learn/vector-database>
- [18] “Retrieval-augmented generation (RAG) — Langchain.” Accessed: Nov. 26, 2023. [Online]. Available: [https://python.langchain.com/docs/use\\_cases/question\\_answering](https://python.langchain.com/docs/use_cases/question_answering)
- [19] “Recursively split by character — Langchain.” Accessed: Nov. 26, 2023. [Online]. Available: [https://python.langchain.com/docs/modules/data\\_connection/document\\_transformers/text\\_splitters/recursive\\_text\\_splitter](https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/recursive_text_splitter)
- [20] “NVD - Home.” Accessed: Oct. 20, 2023. [Online]. Available: <https://nvd.nist.gov>
- [21] “NVD - NVD Dashboard.” Accessed: Oct. 20, 2023. [Online]. Available: <https://nvd.nist.gov/general/nvd-dashboard>

- [22] “CVE security vulnerability database. Security vulnerabilities, exploits, references and more.” Accessed: Dec. 10, 2023. [Online]. Available: <https://www.cvedetails.com/index.php>
- [23] A. Kumar and I. Sharma, “Identifying Patterns in Common Vulnerabilities and Exposures Databases with Exploratory Data Analysis,” in 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Dec. 2022, pp. 919–924. doi: 10.1109/ICACRS55517.2022.10029077.
- [24] M. A. Williams, S. Dey, R. C. Barranco, S. M. Naim, M. S. Hossain, and M. Akbar, “Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database,” in 2018 IEEE International Conference on Big Data (Big Data), Dec. 2018, pp. 3011–3020. doi: 10.1109/BigData.2018.8622299.
- [25] “KeyBERT.” Accessed: Dec. 1, 2023. [Online]. Available: <https://maartengr.github.io/KeyBERT/>
- [26] “You Need To Start Using JSONL In Your Golang Projects!” Accessed: Dec. 10, 2023. [Online]. Available: <https://thegodev.com/handling-jsonl/>
- [27] “Understanding indexes,” Pinecone. Accessed: Dec. 10, 2023. [Online]. Available: <https://docs.pinecone.io/docs/indexes>
- [28] “Pretrained Models — Sentence-Transformers documentation.” Accessed: Nov. 28, 2023. [Online]. Available: [https://www.sbert.net/docs/pretrained\\_models.html](https://www.sbert.net/docs/pretrained_models.html)
- [29] W. Zheng, Z. Chen, X. Wu, W. Fu, B. Sun, and J. Cheng, “A Domain Knowledge-Guided Lightweight Approach for Security Bug Reports Prediction,” in 2021 8th International Conference on Dependable Systems and Their Applications (DSA), Aug. 2021, pp. 359–368. doi: 10.1109/DSA52907.2021.00055.
- [30] S. Ruwanpura, C. Morash, M. A. Khan, A. Ahmad, and G. Ginde, “Automatic Domain-Specific Corpora Generation from Wikipedia - A Replication Study,” in 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), Sep. 2023, pp. 85–94. doi: 10.1109/REW57809.2023.00022.
- [31] B. Issa, M. B. Jasser, H. N. Chua, and M. Hamzah, “A Comparative Study on Embedding Models for Keyword Extraction Using KeyBERT Method,” in 2023 IEEE 13th International Conference on System Engineering and Technology (ICSET), Oct. 2023, pp. 40–45. doi: 10.1109/ICSET59111.2023.10295108.

- [32] “IEEE Xplore.” Accessed: Dec. 10, 2023. [Online]. Available: <https://ieeexplore.ieee.org/Xplore/home.jsp>
- [33] “Home Page — CISA.” Accessed: Dec. 10, 2023. [Online]. Available: <https://www.cisa.gov/>
- [34] “IBM Documentation.” Accessed: Dec. 10, 2023. [Online]. Available: <https://www.ibm.com/docs/en/was/8.5.5?topic=wsspmica-security-considerations-web-services-1>
- [35] “Explore AI models in Model Garden — Vertex AI,” Google Cloud. Accessed: Oct. 12, 2023. [Online]. Available: <https://cloud.google.com/vertex-ai/docs/start/explore-models>
- [36] “Measuring Hallucinations in RAG Systems – Vectara.” Accessed: Dec. 07, 2023. [Online]. Available: <https://vectara.com/measuring-hallucinations-in-rag-systems/>
- [37] N. Badyal, D. Jacoby, and Y. Coady, “Intentional Biases in LLM Responses,” in 2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Oct. 2023, pp. 0502–0506. doi: 10.1109/UEMCON59035.2023.10316060.
- [38] S. Jha, S. K. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, and S. Neema, “De-hallucinating Large Language Models Using Formal Methods Guided Iterative Prompting,” in 2023 IEEE International Conference on Assured Autonomy (ICAA), Jun. 2023, pp. 149–152. doi: 10.1109/ICAA58325.2023.00029.
- [39] R. Feldt, S. Kang, J. Yoon, and S. Yoo, “Towards Autonomous Testing Agents via Conversational Large Language Models,” in 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Sep. 2023, pp. 1688–1693. doi: 10.1109/ASE56229.2023.00148.
- [40] H. Brown, K. Lee, F. Mireshghallah, R. Shokri, and F. Tramèr, “What Does it Mean for a Language Model to Preserve Privacy?,” in 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul Republic of Korea: ACM, Jun. 2022, pp. 2280–2292. doi: 10.1145/3531146.3534642.
- [41] J. Kaplan et al., “Scaling Laws for Neural Language Models.” arXiv, Jan. 22, 2020. doi: 10.48550/arXiv.2001.08361.
- [42] S. Parkinson, P. Ward, K. Wilson, and J. Miller, “Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 11, pp. 2898–2915, Nov. 2017, doi: 10.1109/TITS.2017.2665968.