From Manual to Automated Cyber Risk Assessment: LLM- and RAG-Driven Multi-Agent Threat Modeling with CORAS in Healthcare Case Studies

Gencer Erdogan[1,*], Morgan Gillette[2], Simeon Tverdal[1], and Ragnhild Halvorsrud[1]
[1]Sustainable Communication Technologies, SINTEF Digital, Oslo, Norway
[2]National Graduate School of Engineering and Research Center, Caen, France
{gencer.erdogan, simeon.tverdal, ragnhild.halvorsrud}@sintef.no, morgan.gillette@ecole.ensicaen.fr
*corresponding author

Abstract—Cyber risk assessment is a cornerstone of cybersecurity management, yet current practices remain largely manual, static, and resource intensive. This paper presents the CORAS Threat Modeler, an open-source tool that leverages large language models (LLMs), retrieval-augmented generation (RAG), and multi-agent orchestration to automate the generation of structured risk information and threat models directly from natural-language system descriptions. The tool was developed with three success criteria in mind: automating threat model creation, enabling dynamic risk assessment through context-aware retrieval and generation, and supporting accessibility for both experts and non-experts. We present the architecture of the tool and its integration with CAPEC and CWE repositories, and report on an evaluation across three healthcare case studies: one hospital and two medical device manufacturers. Results show that the tool successfully produced syntactically correct and interpretable threat models, generated contextually relevant risks and mitigations, and lowered the entry barrier for non-experts. However, improvements for broader risk coverage, treatment flexibility, and enhanced usability are needed to fully realize its potential.

Keywords–Cybersecurity; cyber risk assessment; large language models (LLMs); retrieval-augmented generation (RAG); automated threat modeling; healthcare cybersecurity;

1. Introduction

Cyber risk assessment, i.e., the activities of identifying, analyzing, and evaluating security risks affecting the assets of a system or organization [1], remains one of the most important practices for managing security risks and ensuring resilience. However, traditional risk assessment methods are manual, static and resource intensive, relying heavily on expert knowledge and lengthy documentation or audit processes. This approach is often too slow and inconsistent to keep up with the rapidly changing digital environment, where new vulnerabilities and attack vectors emerge daily.

Current approaches to cyber risk assessment struggle to support the demands of dynamic and adaptive cybersecurity practices. Both our previous systematic mapping studies [2], [3], and subsequent work in [4] underline that dynamic and automated risk assessment remains an open challenge, with most existing methods unable to respond in real time to changing vulnerabilities and attack patterns. Moreover, the systematic review in [5] emphasize that emerging technologies such as AI and machine learning are increasingly viewed as critical enablers for automating and accelerating core cybersecurity tasks, including intrusion detection, vulnerability analysis, and risk assessment. Thus, there is a pressing need for risk assessment solutions that move beyond static expert-driven methods and towards automation.

To address this need, we have developed a tool that integrates LLMs to automate the generation of CORAS threat models [6] and reports from natural language descriptions of the target of analysis. CORAS provides a graphical notation to model security risks including threat scenarios, vulnerabilities, unwanted incidents, assets, and countermeasures, thereby facilitating communication between technical experts and decision-makers. While CORAS is mature and widely adopted [7], and has empirically been shown to be intuitive [8], producing such models remains a demanding manual task requiring training and expertise. This limits its ability to support fast or iterative assessments in dynamic contexts.

We refer to our tool as CORAS Threat Modeler, which is open source and freely available [9]. We defined the following success criteria (SC) to guide its development:

- SC1: Automate threat model creation by generating CORAS threat diagrams and structured risk reports directly from natural-language system descriptions.
- SC2: Enable dynamic risk assessment by using LLMs to suggest threat scenarios and mitigations through similarity search and RAG techniques.
- SC3: Support accessibility by enabling both domain experts without prior modeling expertise and non-experts to conduct structured risk assessments.

This paper makes three main contributions:

- We present the design and implementation of the open-source CORAS Threat Modeler, which employs a multi-agent pipeline combining LLMs and RAG to generate threat models represented as directed acyclic graphs.
- The tool integrates the cybersecurity repositories Common Attack Pattern Enumeration and Classification (CAPEC) and Common Weakness Enumeration (CWE), and employs schema-constrained generation to ensure syntactically valid and semantically consistent CORAS threat diagrams.
- We report on an initial evaluation with three healthcare case studies, namely one hospital and two medical device

manufacturers, conducted with target users.

The results demonstrate that the tool can generate syntactically correct and interpretable threat models, broaden practitioners' perspectives on risks, and lower the entry barrier for non-experts, while also pointing to areas for improvement with respect to risk coverage, treatment flexibility, and usability.

The remainder of the paper is structured as follows. Section 2 describes our research method. Section 3 describes the tool design and implementation. Section 4 reports on evaluation experiments with three companies from the healthcare domain: one hospital and two medical device manufacturers. Section 5 discusses the evaluation results with target users, highlighting both the strengths and limitations of the tool. Section 6 reviews related work on LLMs for cyber risk management and threat modeling. Section 7 concludes the paper.

2. Research Method

We carried out the work in a three-step design science approach [10]. In Step 1, we defined the success criteria (SC1–SC3) for the CORAS Threat Modeler based on the background and needs outlined in Section 1. These criteria capture what the tool should achieve in terms of automating threat model creation, enabling dynamic risk assessment, and supporting accessibility for both experts and non-experts.

In Step 2, we designed and implemented the tool in accordance with the success criteria. The design process combined the CORAS methodology with state-of-the-art advances in LLMs, RAG, and multi-agent orchestration. The implementation includes the component architecture, interaction flow, and integration with cybersecurity knowledge bases such as CAPEC and CWE. The design and implementation are presented in detail in Section 3.

Finally, in Step 3 we evaluated the tool together with target users from three healthcare case studies: one hospital and two medical device manufacturers. The evaluation process was structured around the success criteria and consisted of three phases: preparation, workshop execution, and follow-up. In the preparation phase, we developed case descriptions based on input from the case study providers and verified them together. In the workshop phase, participants interacted with the tool while answering 21 targeted questions about the generated outputs. In the follow-up phase, participants were asked to reflect on their overall impressions and add any feedback not captured during the workshops. The evaluation procedure and results are described in Section 4.

3. Implementation

In the following, we first introduce the CORAS risk assessment method and then describe how the CORAS Threat Modeler components interact to automatically generate structured risk information and threat models from natural-language descriptions of a target system.

CORAS is a model-based method for asset-driven, defensive cyber risk assessment [6]. The method comes with detailed guidelines explaining how to conduct each stage of a CORAS risk assessment in practice. CORAS is aligned with ISO 27005 [1] and consists of eight steps covering the full process from initial preparations to the derivation of risks and risk treatment recommendations. To support this process, the CORAS modeling tool facilitates the creation of five types of diagrams, designed to support specific stages in the risk assessment. These diagrams help users model and reason about threat actors, threat scenarios, vulnerabilities, unwanted incidents, assets, and risk treatments.

In this paper, we focus on the CORAS tool and present our enhancements that enable automatic and dynamic risk assessment through LLM-based multi-agent support with RAG (the CORAS Threat Modeler), as mentioned in Section 1. For a detailed explanation of the CORAS method and its steps, as well as its calculus for risk estimation and the formal syntax and semantics of the CORAS modeling language, we refer the reader to the CORAS book [6].

Figure 1 illustrates the implementation of the CORAS Threat Modeler through a sequence diagram. The tool consists of a client-side graphical user interface and multiple server-side components that interact to generate: (i) structured case descriptions from the user-provided free-text system description (target of analysis), (ii) high-level risk tables that capture threat scenarios and risks, and (iii) CORAS threat models generated from the structured risk information in point (ii). In the following we describe the interaction flow between the tool components represented by the lifelines in Figure 1.

The graphical user interface (*:GUI*) is the entry point for the user. It allows users to input natural-language free-text descriptions of their system (target of analysis) and interact with the generated outputs, including the structured case descriptions, high-level risk tables, and CORAS threat models. The GUI runs client-side in the browser and communicates with the back-end via HTTP requests.

The Flask App (*:FlaskApp*) serves as the API entry point on the back-end. It listens for incoming requests from the GUI, maps them to predefined routes, and invokes the corresponding back-end functions. The functions *generate_summary()*, *generate_risks()*, and *generate_coras_model()* generate structured case descriptions, high-level risk tables, and CORAS threat models, respectively. The Flask App parses JavaScript Object Notation (JSON) input, handles the request, calls the Navigator, and returns JSON responses to the client.

The Navigator (*:Navigator*) acts as the central controller and provides a single entry point for the workflow. Rather than the client or Flask App interacting with each agent directly, the Navigator orchestrates the overall process by invoking the appropriate specialized agents in sequence. For example, it calls the Summarizer agent to produce the structured case description, the RAG module and Assessor agent to generate the high-level risk tables, and finally the Formatter agent to construct the CORAS threat model. This approach hides underlying complexity and ensures a consistent workflow.

The Summarizer agent (*:Summarizer*) transforms the free-text system description provided by the user into a structured description. It uses the `llama3:70b-instruct` model with temperature set to 0 and a fixed summarization prompt.

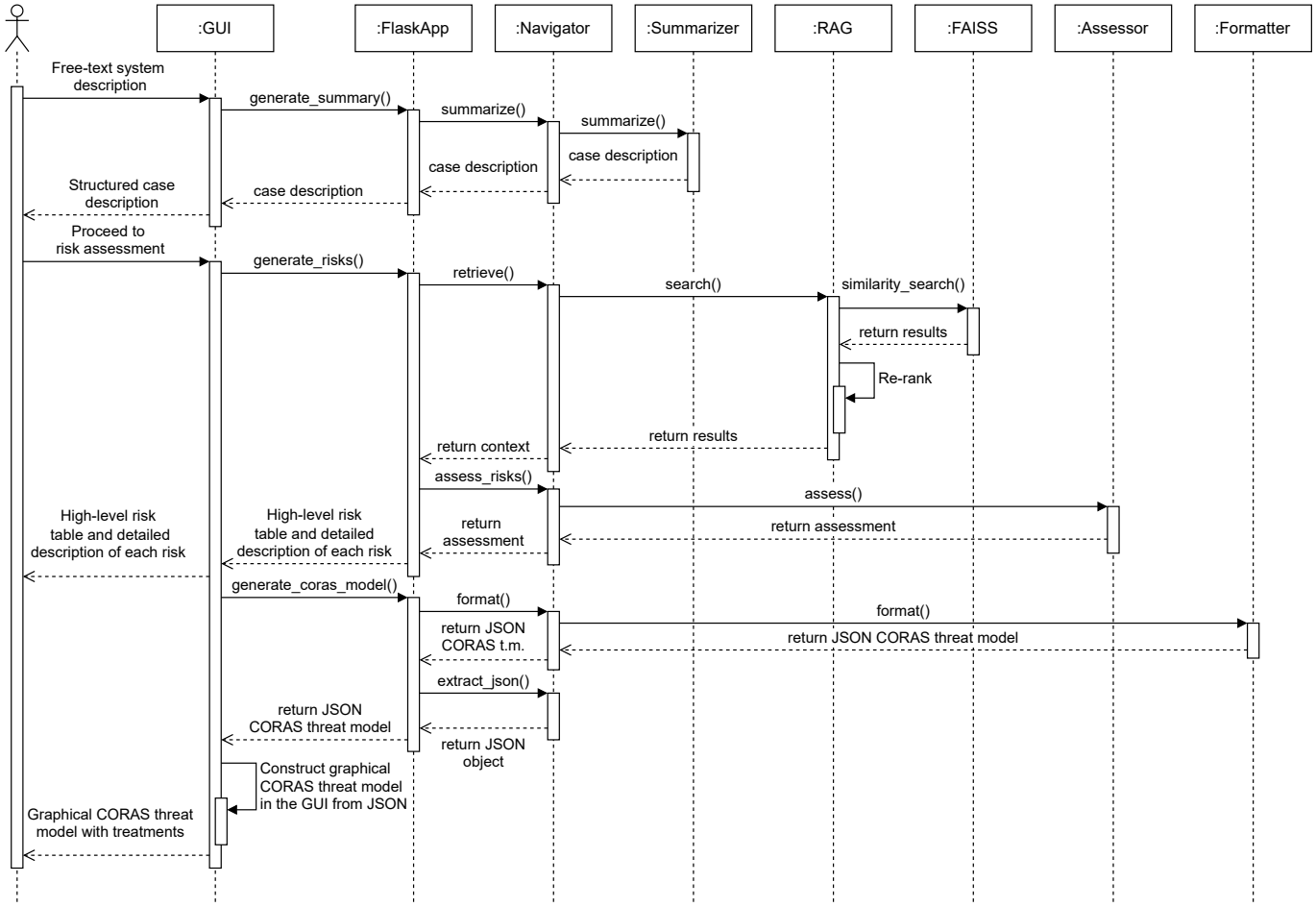

Figure 1. Interaction flow of CORAS Threat Modeler components.

LangChain (a framework for building agents and LLM-powered applications) is used to manage prompt templates and execution. The Summarizer ensures that the system description is reformulated in a clear and structured way, suitable as input for later analysis.

For risk assessment, we retrieve relevant cybersecurity threat context from the knowledge bases CAPEC and CWE. The Retrieval-Augmented Generation module (*:RAG*) embeds structured case descriptions into a Facebook AI Similarity Search (*:FAISS*) vector database and performs similarity search to identify relevant entries from CAPEC and CWE. As shown in (1), the similarity search returns the index j corresponding to the vector x_j that has the minimum Euclidean distance $\|x - x_i\|$ to the input vector x:

$$j = \arg\min_i \|x - x_i\| \tag{1}$$

Although vector similarity search captures semantic relatedness, it does not always guarantee that the retrieved entries are the most contextually relevant to the system (target of analysis) [11]. To address this, we apply a re-ranking step that evaluates the top-k retrieved results and selects the subset most closely aligned with the user query system description, ensuring that only contextually appropriate CAPEC and CWE entries are passed to the Assessor. This provides the Assessor with contextualized knowledge of vulnerabilities and attack patterns. For the re-ranking, the RAG module uses the smaller `llama3:8b` model, which offers a good balance between efficiency and accuracy for this filtering task.

The Assessor agent (*:Assessor*) generates a textual risk assessment by combining the structured case description with the retrieved context. It uses the `llama3:70b-instruct` model (temperature 0) with a dedicated system prompt that enforces a two-phase assessment. In the first phase, it produces a high-level risk table that lists each identified risk in terms of the threat actor (e.g., insider, attacker), threat scenarios (e.g., malware injection, unauthorized access), vulnerabilities (e.g., CWE-311 missing encryption of sensitive data), unwanted incidents (e.g., data leakage), and impacted assets (e.g., patient data, medical device). In the second phase, it elaborates on each risk element, providing detailed descriptions of the chain of threat scenarios, which vulnerabilities are involved, and what mitigations may apply. The resulting assessment provides detailed risk information that serves as input for the construction of the CORAS threat model.

The Formatter agent (*:Formatter*) converts the textual risk assessment produced by the Assessor into a machine-readable JSON object that defines a syntactically correct CORAS threat model. Because a CORAS threat model is a directed acyclic graph, the JSON format consists of two main elements: a list of vertices and a list of edges. Vertices represent the core CORAS concepts and are typed as either human threats (malicious or non-malicious), non-human threats, threat scenarios, unwanted incidents, assets, or mitigations. Each vertex has a unique identifier and descriptive text. Vulnerabilities may be assigned to edges (relations), describing a weakness, flaw, or deficiency (e.g., a CWE entry), ensuring that causal relations between threat actors, threat scenarios, and unwanted incidents are explicitly captured (see Figure 3).

To ensure valid and correctly structured outputs, the Formatter uses structured output techniques that constrain the LLM (`llama3:70b-instruct` model and temperature 0) to generate JSON objects compliant with a predefined schema. This guarantees syntactic validity and prevents the need for extensive error handling. In addition, few-shot prompting with worked examples is employed to guide the LLM toward producing outputs consistent with CORAS semantics, and explicit constraints on vertex and edge types further reduce invalid structures. Post-processing steps are applied to improve the quality of the model, such as removing duplicate threat scenarios generated during the assessment.

Finally, the processed JSON object is rendered into a visual CORAS threat diagram in the GUI. Rendering is performed using JointJS, where the graph elements are positioned by grouping vertices into hierarchical levels with a breadth-first search (BFS) algorithm and placing them at predefined coordinates. This ensures that the resulting diagram is not only syntactically correct but also visually consistent and interpretable for the user.

The tool's source code, including key prompts and JSON schemas, is openly available in the CORAS GitHub repository [9], and its components form a modular pipeline that transforms natural-language system descriptions into structured risk information and CORAS threat models, thereby operationalizing the success criteria introduced in Section 1.

4. Evaluation

We evaluated the CORAS Threat Modeler with three companies from the healthcare domain: one hospital and two medical device manufacturers. The authors and the three companies are partners in the EU-funded NEMECYS project [12], which develops tools and methods to help manufacturers, integrators, and healthcare providers ensure cybersecurity by design for connected medical and diagnostic devices (CMDs). Each company provided its own case, resulting in three distinct case studies. The evaluation consisted of three phases: preparation, workshop execution, and follow-up, as illustrated in Figure 2.

4.1 Preparation

For each company, we used their free-text system descriptions as documented in the publicly available report *D4.1 Validation of all Case Studies*, available on the NEMECYS project website [12]. The free-text system descriptions were sent by email to the respective case providers for review and correction. After making any requested adjustments, the partners confirmed that the descriptions were accurate and representative of their systems. The confirmed descriptions were then used during the workshop execution as input to the CORAS Threat Modeler. For brevity, we provide only a short overview of each case study. Full descriptions are available in the publicly available report mentioned above.

Case Study 1 (Hospital Point-of-Care Testing): This case study concerns the use of a Continuous Glucose Monitoring kit at San Raffaele Hospital (OSR) in Milan. The system involves patients who wear a subcutaneous glucose sensor on the upper arm and health professionals who access the collected data for treatment and monitoring. Glucose readings are obtained either via a dedicated Abbott Reader or through the FreeStyle LibreLink smartphone application, with data further shared to the LibreView web application for clinical use. While the sensor communicates securely through encrypted wireless channels, two points of concern were identified in the hospital setting: (i) access to the LibreView account, which depends on the robustness of authentication and provider-side security updates, and (ii) connecting the Abbott Reader to hospital computers via USB, which raises the risk of malware transfer in either direction. In the latter, a patient brings the reader to a hospital consultation, and the diabetologist connects it to an OSR computer to transfer and analyze glucose data using a smart digital clinic software.

Case Study 2 (Wearable Monitoring for Parkinson's Disease): This case study, provided by PD Neurotechnology (PDN), addresses the cybersecurity of the PDMonitor, an IoT-based wearable medical device designed for the continuous monitoring of Parkinson's disease patients in both hospital and home settings. The system comprises five Inertial Measurement Unit sensors worn by the patient, a docking station for local data collection and transfer, a mobile application for patient and caregiver interaction, a cloud infrastructure for storage and machine learning–based analysis, and a web dashboard for physician access. In this scenario, the device also integrates with Deep Brain Stimulation (DBS) therapy by generating updated DBS settings and transmitting them to the implant via the mobile app. Typical use involves patients

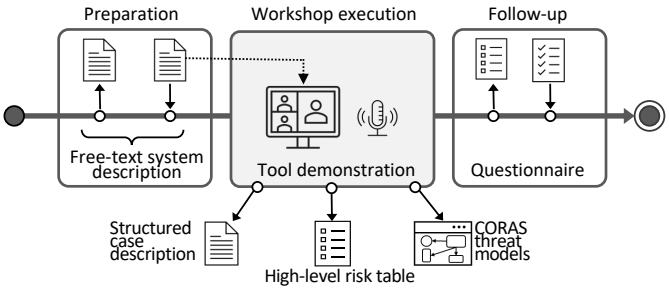

Figure 2. Overview of the evaluation procedure.

wearing the sensors in daily life, with data uploaded through the docking station to the cloud, where advanced analytics generate symptom reports for physicians and DBS parameter adjustments. The case highlights cybersecurity risks across the ecosystem, particularly in cloud connectivity, device–cloud communication, and Bluetooth-based transmission to the DBS implant, all of which are critical points where confidentiality, integrity, and availability must be safeguarded.

Case Study 3 (Home Dialysis Monitoring): This case study, provided by Mode Sensors (MODE), focuses on Re:Balans, a non-invasive wearable bioimpedance device developed to monitor hydration levels in patients with fluid management conditions, such as those requiring home dialysis. The sensor continuously collects hydration data and transmits it wirelessly to external units (e.g., tablets or IoT gateways) for integration with clinical systems and remote patient monitoring. The evaluation highlighted several cybersecurity challenges across the device lifecycle: secure wireless transfer between the patch and gateway, secure transmission of data from the gateway to healthcare systems, reliable and secure software updates, post-market surveillance to ensure intended use, and mechanisms for user feedback without compromising confidentiality. The high-level architecture of the device comprises a Bluetooth radio, debug interface, microcontroller, external flash, and multiple I/O interfaces. This case underlines the importance of robust security guidelines for data flows beyond the hospital setting, ensuring safe connectivity, software maintenance, and feedback processes in remote care scenarios.

4.2 Workshop Execution

Workshops were conducted via online meetings, one for each case study. Each session followed a structured format. First, participants were given a short introductory presentation covering three points: 1) the motivation and objectives of the CORAS Threat Modeler, 2) an explanation of CORAS threat modeling concepts and notation, and 3) a set of ground rules, emphasizing that the evaluation focused on the tool rather than the participants' expertise or organization, and that both positive and critical feedback was valuable.

After the presentation, one of the authors operated the CORAS Threat Modeler on their own computer and shared the screen with the participants. The tool was executed on an HP Z8 Fury G5 workstation equipped with a 36-core Xeon processor, 512 GB RAM, and dual NVIDIA RTX 6000 Ada GPUs. The case descriptions from the preparation phase were then used as inputs to the tool, which was run step by step. During this process, we asked participants targeted questions related to the outputs produced. The full set of questions is provided in Table II: Questions 1–3 addressed the *structured case description* (first output), Questions 4–9 the *high-level risk table and detailed description of each risk* (second output), Questions 10–15 the generated *CORAS threat model* (third output), and Questions 16–21 their overall impressions.

Table I summarizes the workshops, showing the meeting dates, case study (CS), number of participants from the case study provider (Pts.), self-assessed risk assessment expertise (RAE), self-assessed CORAS expertise (COE), and job titles. The self-assessments were made by the participants from the case study provider using a four-level Likert scale {Novice, Beginner, Intermediate, Expert}. For Case Study 1, two service designers participated; for Case Study 2, one information security officer (ISO); and for Case Study 3, one chief technical officer (CTO). Each workshop lasted approximately two hours, with two of the authors present in all meetings to facilitate the sessions. The sessions were recorded, with both video and automatically generated transcriptions of the discussions.

4.3 Follow-up

After each workshop, participants received a follow-up email to ensure that no feedback was overlooked and to allow them to share additional reflections after the session. The email included the generated workshop report, which contained the three outputs produced during the session: the structured case description, the high-level risk table and detailed description of each risk, and the generated CORAS threat model. Participants were asked to revisit Questions 16–21 from Table II, which concerned their overall impressions of the tool.

They were encouraged to provide short written responses or bullet points. This follow-up process ensured that additional insights could be captured beyond the live discussions, and the written responses were integrated with the feedback already collected during the workshops.

4.4 Results

Table II provides a concise summary of the participants' answers to the 21 evaluation questions across all three case studies. The first output, structured case descriptions, was deemed accurate and satisfactory across all cases (Q1). Nothing essential was missing, although the participant in Case Study 1 noted that the original case description omitted aspects related to physical tampering (Q2). However, this omission was due to the fact that such aspects were not considered in their case study and were therefore out of scope. No unnecessary or irrelevant content was reported by the participants in the structured case descriptions (Q3).

The high-level risk tables contained relevant scenarios, including malware transfer and unauthorized actors, which in one case were highlighted as risks not previously considered, alongside other realistic threats (Q4). The high-level risk tables were generally considered sensible, accurate, and useful, though one participant noted that it was unclear which component in their case a given threat applied to, and another pointed out that the mitigations were broad but still appropriate (Q5). Participants partly expected to see the identified threat scenarios, though one found the level of detail higher than

TABLE I
WORKSHOP MEETINGS AND PARTICIPANTS

Date	CS	Pts.	RAE	COE	Job Title
01.09.2025	1	2 Pts.	Beginner	Beginner	Service Design
02.09.2025	2	1 Pts.	Intermediate	Novice	ISO
10.09.2025	3	1 Pts.	Beginner	Novice	CTO

TABLE II
CONCISE Q&A SUMMARY FROM CASE STUDY 1, 2, AND 3 (BASED ON TRANSCRIPT EXCERPTS AND FOLLOW-UP FEEDBACK)

Q	Question	Case Study 1 - OSR	Case Study 2 - PDN	Case Study 3 - MODE
Q1	Are you satisfied with the structured description? Is it Accurate?	Yes–satisfactory and accurate; nothing missing.	Yes–accurate, well grouped; avoided irrelevant details.	Yes–OK/accurate; scope biased by prior security focus.
Q2	Is something missing in the structured description?	No–nothing missing.	No–nothing missing given well-structured input.	Nothing missing in output; input itself omitted parts.
Q3	Is anything included that should not have been?	No.	No–nothing extra; all listed correctly.	No.
Q4	Does the high-level risk table contain relevant threat scenario(s)?	Yes–especially malware transfer and "unauthorised actor".	Yes–both relevant and exist.	Yes–both scenarios relevant.
Q5	Does the high-level risk table makes sense? Is it Accurate?	Mostly yes–useful new risk; some needed info missing/unclear.	Yes–sensible, accurate, useful at brainstorming; mitigations general (appropriately).	Yes–makes sense; details OK/accurate.
Q6	Did you expected to see these threat scenarios?	Partly–expected in those lines, but not at that detail level.	Yes–matches top risks; expecting priority on patient health.	Yes–obvious/known; wanted less-obvious additions.
Q7	Are you missing any threat scenarios?	No.	Yes–many missing; especially health-impacting risks.	Yes–only two shown; expects a longer list with selection step.
Q8	Is the risk table's level of abstraction appropriate?	Good–detailed and complete enough.	Just right–balanced for brainstorming/usability.	Fine for two items; with more, start high-level then expand; list many vulnerabilities.
Q9	Is the terminology understandable?	Yes.	Yes–clear; minor quibble with "unwanted incident" term (not important).	Yes–high-level table more readable; detailed information suits implementers.
Q10	Is the CORAS threat model easy to understand?	Yes–for non-experts; some technical parts may be hard.	Yes–easy; even clearer with intro material.	Yes–easy to read/understand.
Q11	Is the threat model more useful than the table for understanding risks?	Yes–clearer, easier for non-experts than table alone.	For simple cases little added; for complex, clearly beneficial.	Yes–easier to see event sequences/what leads to what.
Q12	Do you have any comments about the graphical notation?	Yes–inconsistency in vulnerability labeling (IDs vs text).	No comments; suggests modernizing icons (assets).	No major issues; asked about meaning of crossed arrows.
Q13	Does the threat model include the expected information?	Yes.	Yes–to the point for brainstorming; avoids excess detail.	Yes–contains expected info; OK.
Q14	Is the threat model's level of abstraction useful?	Yes–detailed/complete enough.	Yes–ideal balance for human-centric brainstorming.	Yes–enough information; not too much.
Q15	Is this way of generating CORAS threat models useful for you?	Yes–useful, especially in hospital context.	Yes–brainstorming, mgmt. presentations, onboarding/training.	Yes–identifies vulnerabilities and sensible fixes.
Q16	What is your overall impression?	Positive–generated report is clear/useful for management.	Positive–generally happy with the tool.	Positive–wants ability to steer and more risks to pick from.
Q17	Did you notice any issues with the user interface or usability?	Yes–too much scrolling; prefer step-by-step with progress/confirmation.	Yes–vulnerabilities appear disconnected in the threat model.	Yes–User interface sparse; needs polish and suggestions from a user experience expert.
Q18	Would you like to use such a tool?	Yes–useful for hospital; report helpful.	Yes.	Yes–sees practical value for securing their system.
Q19	In what situations could this tool be useful in your work?	Management discussions; support security investment decisions; for cybersecurity dept.	Early brainstorming; management presentations; onboarding/training.	Across phases—analysis through dev/prod; prioritise security effort; cross-org use.
Q20	In what situations could this tool be useful for others?	Management and cybersecurity staff; review risk treatments.	Brainstorming, presentations, onboarding, awareness/training.	Developers and cybersecurity staff.
Q21	What are the main areas for improvement?	Clearer vulnerability info (descriptions/links, not just CWE IDs); stepwise navigation;	Treatments should be less prescriptive, more generic/options with disclaimers; add API.	User interface: too sparse; needs polish and user experience input.

anticipated, another expected stronger focus on patient health (i.e., physical harm caused by cyber risks), and a third wanted less-obvious additions to the list of potential risks (Q6). Regarding missing threat scenarios, one participant saw none, while the others found many absent, especially health related risks, and expected a longer list with a selection step to give users more control over which risks to focus on in the following tool steps (Q7). The level of abstraction was deemed appropriate, with comments ranging from detailed and complete enough, to balanced for brainstorming, to a suggestion to start high level and then expand into the risks most relevant for the user (Q8). Terminology was broadly understandable, with only minor comments, such as a remark that the term "unwanted incident" is redundant since an incident already implies something undesirable, while also noting that different standards (e.g., ISO 27005 [1]) use varying terms. Another observation was that high level tables were more readable, whereas the detailed information suited implementers (Q9).

The CORAS threat models were considered easy to understand, though one participant mentioned that technical parts may be harder for non-experts (Q10). Figure 3 shows the CORAS threat model automatically generated from the free-text system description of Case Study 3. The model contains two chains of threat scenarios. In the upper chain, a deliberate human threat, *External attacker*, initiates the threat scenario *Attacker intercepts wireless transmission between Re:Balans device and external unit*. This may lead to the threat scenario *Attacker exploits vulnerabilities in encryption protocols to*

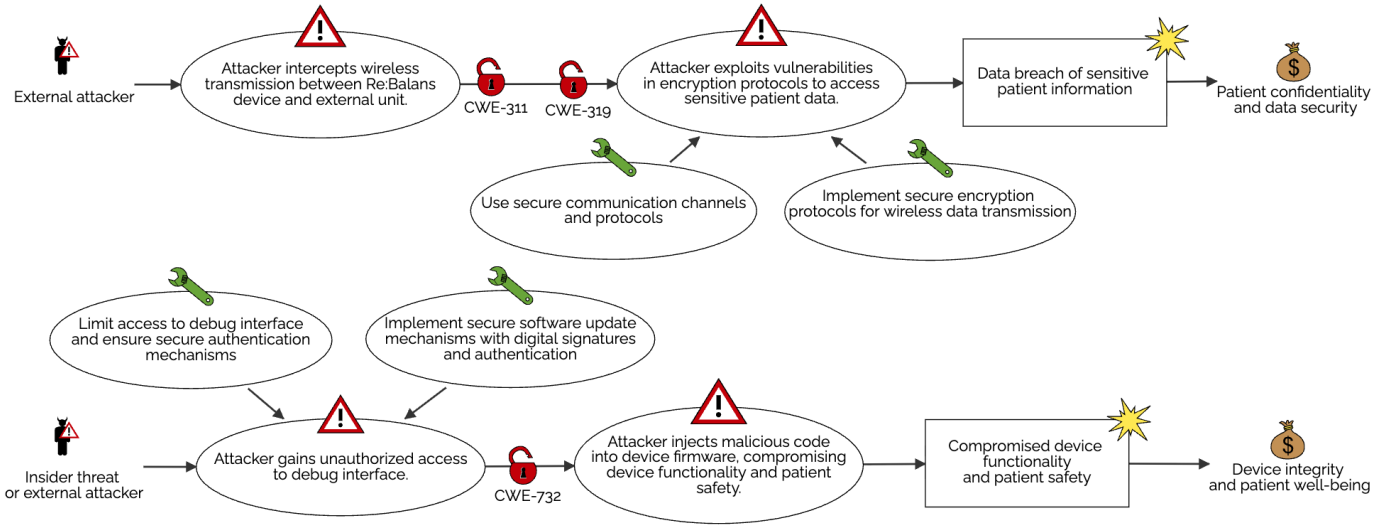

Figure 3. Automatically generated CORAS threat model for Case Study 3.

access sensitive patient data, through exploitation of *CWE-311: Missing Encryption of Sensitive Data* and *CWE-319: Cleartext Transmission of Sensitive Information*. This threat scenario, in turn, leads to the unwanted incident *Data breach of sensitive patient information*, which impacts the asset *Patient confidentiality and data security*. The tool also suggested two treatments for the encryption-related threat scenario: *Use secure communication channels and protocols* and *Implement secure encryption protocols for wireless data transmission*.

Compared to the tables, the CORAS threat models were generally seen as more useful, since they made sequences and dependencies easier to follow, although one participant felt the added value was clearer in complex cases than in simple ones (Q11). Comments on the notation included an inconsistency in labeling vulnerabilities, where some were shown only with CWE IDs while others included descriptive text. Although this information was consistently present in the high level risk table, it was not always reflected in the threat models. One participant suggested modernizing icons, noting that alternatives to symbols such as money bags could better represent assets, even if the current representation was a conscious design decision for CORAS and empirically shown to be comprehensive [8]. Another comment concerned two arrows that happened to cross unnecessarily in one model; the participant initially assumed this was deliberate or had a semantic meaning (Q12). The models were reported to include the expected information and to the right level of detail, avoiding unnecessary complexity (Q13 and Q14). The way of generating CORAS threat models in our approach was considered useful across cases, with applications ranging from hospital use and medical device design to early brainstorming, management presentations, onboarding, training, and identifying vulnerabilities and fixes (Q15).

Overall impressions were positive, with participants describing the tool as clear, useful, and a good approach, while also noting the importance of being able to steer the outputs and expand them with more risks to choose from (Q16). Usability issues included too much scrolling, since each step of the tool produced results vertically rather than through a step-by-step progress with confirmation. Vulnerabilities appeared visually disconnected because, although attached to relations in CORAS, this was not explicitly shown with connecting lines. Further comments concerned the sparse interface design and the need for UX improvements (Q17). All expressed interest in using the tool themselves, seeing practical value for hospitals, medical device manufacturers, security work, and securing systems (Q18). Participants suggested usage situations in which CORAS Threat Modeler could be useful in their work, including management discussions, supporting security investment, early brainstorming, onboarding, awareness training, and cross-phase use from analysis through production (Q19). Potential users beyond their own work included management, cybersecurity staff, developers, and trainers (Q20). Suggested improvements covered clearer vulnerability information (with better descriptions and links to the CAPEC and CWE repositories), less prescriptive treatments with options and disclaimers, programmatic API access, and an overall more polished and user-friendly interface (Q21).

5. Discussion

In this section, we discuss the evaluation results with respect to the success criteria defined in Section 1.

The first success criterion (SC1) required the tool to automate threat model creation by generating CORAS threat diagrams and structured risk reports directly from natural-language system descriptions. The evaluation results show that this criterion was largely achieved. Structured case descriptions were deemed accurate and satisfactory across all cases, with no unnecessary information included by the agentic modules (i.e., *:Summarizer*, *:RAG*, *:Assessor*, and *:Formatter*), and only minor omissions that stemmed from the scope of the

case descriptions rather than from the tool itself. The high-level risk tables and the generated CORAS threat models were also considered sensible and accurate. Participants noted that these outputs supported brainstorming. They also found them useful as a basis for management discussions. From a technical perspective, the tool successfully generated syntactic and semantically correct CORAS threat models in all cases.

In terms of limitations, one case highlighted that it was unclear which system component from the free-text description a specific threat scenario applied to, and inconsistencies were observed in vulnerability naming, with some shown only as CWE identifiers and others with descriptive text. These findings suggest that SC1 is met in principle, though refinements are needed to improve clarity and consistency of the generated outputs. Importantly, the threat models were found easier to interpret than the risk tables for understanding causal relations and the chain of events, indicating that the automatically generated CORAS threat models provide clear added value for communicating and reasoning about risks.

The second success criterion (SC2) aimed to enable dynamic risk assessment through LLM-driven generation of threat scenarios and mitigations, supported by RAG. The evaluation confirmed that relevant risks were produced and, in some cases, participants identified new risks they had not previously considered, such as malware transfer and unauthorized access. This demonstrates the potential of the tool to broaden practitioners' risk perspectives by identifying risks and attack vectors beyond their immediate concerns, thereby strengthening preparedness against future threats.

However, the coverage of risks was limited. Participants noted that health-related risks, particularly those directly impacting patient safety, were underrepresented, and that the number of scenarios was too few to support selection or prioritization. The underrepresentation of safety-related risks comes from the current reliance on repositories such as CAPEC and CWE, which focus primarily on cybersecurity aspects rather than on the safety impacts of cyber risks. This limitation could be addressed by incorporating additional sources specifically related to cyber-physical security in the healthcare and CMD domain. The restricted number of scenarios was also a deliberate design choice: in this version of the tool, up to the top five risks were generated through similarity search to avoid overwhelming users with excessively large or complex threat models. Finding the right balance between sufficient coverage and manageable visual complexity remains an area for future work. While the level of abstraction was generally deemed appropriate for brainstorming, participants emphasized the need for broader coverage and more control over which risks to focus on in the assessment. Treatments were found useful but sometimes too prescriptive, with a preference for more generic options accompanied by disclaimers. As one participant explained, users should not apply treatments uncritically. Instead, recommendations would be more valuable if presented as options that require expert judgment and adaptation to technical specifications. This is in fact the intended level of abstraction of our risk treatment options, which, as illustrated in Figure 3, are provided at a generic level and require users to adapt and specialize them for their specific context.

The tool demonstrated the capability to generate contextually relevant threat scenarios and mitigations using an LLM- and RAG-driven multi-agent approach. To this end, SC2 has been achieved, though improvements are still needed to broaden coverage, incorporate safety-focused threat scenarios, and provide more flexible treatment recommendations.

The third success criterion (SC3) required the tool to support accessibility by enabling both domain experts without prior modeling expertise and non-experts to conduct structured cyber risk assessments. The evaluation results indicate that this criterion was largely met. All participants, including service designers (Case Study 1), an information security officer (Case Study 2), and a chief technical officer (Case Study 3), were able to engage with the outputs and provide detailed feedback. The structured descriptions, risk tables, and threat models were deemed comprehensible, with the graphical notation in particular making causal relations and chains of events clearer. Participants reported that the tool could be valuable across different contexts, from management presentations and onboarding to detailed security analysis. Several expressed interest in using such a tool themselves, showing that accessibility is not limited to security specialists. However, usability issues were highlighted: excessive scrolling caused by the vertical layout of results, vulnerabilities appearing visually disconnected from relations, and the interface design requiring UX improvements. These findings suggest that SC3 is largely achieved, as the tool lowers the entry barrier to structured cyber risk assessment and enables non-experts to participate meaningfully. Participants also noted that the tool helps overcome the difficulty of starting from a blank page when identifying risks, which is an issue particularly relevant for small and medium enterprises (SMEs) [13]. The ability to quickly generate initial threat models was seen as a valuable basis for subsequent in-depth assessments. Nevertheless, user interface improvements and clearer visualizations are needed to fully realize the tool's accessibility potential.

Across all case studies, participants reported positive impressions and saw practical value in contexts such as management discussions, security investment decisions, onboarding, and awareness-raising. Overall, the results demonstrate that LLM- and RAG-driven multi-agent orchestration can move cyber-risk assessment from manual, resource-intensive processes toward scalable, automated, and accessible tooling, provided that iterative refinements continue to improve coverage, safety alignment, and user experience.

6. Related Work

6.1 LLMs for Cybersecurity and Cyber Risk Management

Reference [14] reviews 185 studies on LLMs in cybersecurity. While current approaches focus on software and network protection, emphasizing detection and repair, our tool leverages LLMs, RAG, and multi-agent orchestration to generate structured risk information and graphical threat models from natural language system descriptions (target of analysis). This

approach directly addresses two of the survey's identified challenges and opportunities: (i) integrating external knowledge repositories (e.g., CAPEC and CWE) to mitigate data scarcity and improve factuality, and (ii) improving accessibility and interpretability by providing visual models that clarify causal relations and risk scenarios. Thus, the CORAS Threat Modeler expands the scope of LLM for security from operational detection tasks to strategic decision support in risk management.

As for risk management, several related approaches exist. Reference [15] introduces a framework named aCTIon that implements a two-step pipeline to condense unstructured reports and extract structured cyber threat intelligence in Structured Threat Information eXpression (STIX) format.

Reference [16] introduces CVEDrill, a predictive LLM model that automates the analysis of Common Vulnerabilities and Exposures (CVEs) by estimating Common Vulnerability Scoring System (CVSS) vectors and classifying vulnerabilities into CWE categories. This enables more accurate vulnerability prioritization and mitigation compared to manual CVE handling.

Reference [17] proposes to use Chain-of-Thought (CoT) prompting with LLMs to support cybersecurity risk assessments for embedded operational technology (OT) systems. In their study, threats from MITRE EMB3D are assessed step by step, including impact, exposure, vulnerability, and likelihood, following IEC 62443 guidance.

These approaches demonstrate the growing use of LLMs to structure and enrich cybersecurity knowledge, from CTI extraction and vulnerability scoring to reasoning about embedded-system risks. The CORAS Threat Modeler complements these approaches by integrating such structured knowledge sources into higher-level threat models that link technical threats and vulnerabilities to organizational risk scenarios.

6.2 LLMs for Threat Modeling

Reference [18] proposes a method that uses retriever-augmented LLMs to automate attack-graph generation. The method builds a semantic CVE database and uses a tailored retriever to fetch contextually relevant CVEs, then prompts an LLM to chain vulnerabilities by matching preconditions and postconditions into attack paths. Unlike CORAS threat modeling, which produces structured chains of threats, scenarios, incidents, and affected assets to support risk assessment, [18] focuses on linking CVEs into attack paths. The produced outputs therefore complement CORAS by providing detailed vulnerability exploitations, but they require mapping and contextualization before they can be used directly for structured risk assessment and treatment planning.

A framework that applies prompt engineering Chain of Thought (CoT), Optimization by PROmpting (OPRO), and fine-tuning to automate threat modeling for banking systems is proposed in [19]. This approach leverages datasets generated from Microsoft's Threat Modeling Tool and maps threats to mitigations aligned with NIST 800-53 controls. This domain-specific adaptation enables LLMs to reason about banking architectures and generate mitigation strategies more effectively than general-purpose LLMs. While [19] focuses on STRIDE-based threat identification and compliance-driven mitigations in the financial sector, the CORAS Threat Modeler complements this work by producing structured, scenario-driven threat models that explicitly connect threats, vulnerabilities, and incidents to organizational risks supporting contextual reasoning and visualization of threats for decision-making.

Reference [20] proposes an LLM-supported threat modeling framework for transportation cyber-physical systems. This approach combines STRIDE-based threat identification with LLM-assisted mapping to MITRE ATT&CK techniques and CVSS-based prioritization. Threats are represented using data flow diagrams in the Microsoft SDL tool, which emphasize system flows and interactions. Our tool complements this by producing CORAS diagrams, formally structured directed acyclic graphs that connect threats, incidents, and assets to support systematic risk assessment and treatment planning.

Reference [21] proposes an LLM-assisted approach for threat modeling of LLM-integrated applications. The approach uses RAG over design artefacts and prior threat models, applies frameworks like OWASP Top 10 for LLMs and MITRE ATLAS, and represents systems (target of analysis) with data-flow diagrams. Our tool represents risks with CORAS diagrams as directed acyclic graphs, and integrates knowledge bases such as CAPEC and CWE to identify and model threat scenarios based on free-text system descriptions. Moreover, while [21] focuses on LLM-specific software threats, our tool can use this information as an additional source to enrich broader risk assessments with LLM-specific threat scenarios.

Reference [22] proposes an LLM-driven framework for cyber-attack scenario generation that produces three synchronized representations: natural language narratives, attack graphs, and formal mathematical models. Their Tri-Modal approach aims to improve interpretability across both technical and non-technical stakeholders by ensuring semantic consistency in scenario descriptions. Our tool complements [22] by supporting higher-level risk reasoning through CORAS diagrams. While their Tri-Modal outputs primarily facilitate cross-role communication and training exercises, CORAS has already been shown [8], and further confirmed in our evaluation, to support effective threat and risk communication between stakeholders with diverse backgrounds. Our tool extends this by automatically generating CORAS threat models directly from user-provided free-text descriptions of the target system, allowing both experts and non-experts to contribute to structured cyber risk assessments.

7. Conclusion

We introduced the CORAS Threat Modeler, an open-source tool [9] that combines LLMs, RAG, and multi-agent orchestration to automate cyber risk assessment. It generates structured risk information and CORAS threat models from natural-language system descriptions, integrating knowledge from CAPEC and CWE, and uses a schema-constrained generation to ensure syntactic and semantic consistent threat models.

Evaluation across three healthcare case studies, one hospital and two medical device manufacturers, showed that the tool produces accurate and interpretable threat models, broadens practitioners' risk perspectives, and enables guided assessments non-experts can meaningfully engage in. Participants valued the tool's ability to provide rapid first drafts of realistic threat scenarios and treatments, though they noted improvement needs in risk coverage, treatment flexibility, and usability. The results highlight the feasibility of LLM- and RAG-driven multi-agent systems for advancing cyber risk assessment from manual, resource-intensive processes to scalable, automated, and accessible practices. While evaluated in healthcare, the tool is domain-agnostic and applicable to other contexts.

Future work will extend safety-related risk coverage, enhance interaction and visualization, integrate additional RAG knowledge sources, and validate cross-domain generalizability, including quantitative metrics (e.g., precision and recall) to strengthen the evaluation.

Acknowledgment

This work is supported by the EU project NEMECYS (Grant 101094323) and the SINTEF project GUARDIAN (Research Council of Norway basic funding).

References

[1] ISO, "ISO/IEC 27005:2018 - Information technology - Security techniques - Information security risk management," ISO, Standard, 2018.

[2] G. Erdogan, E. Garcia-Ceja, Å. Hugo, P. H. Nguyen, and S. Sen, "A Systematic Mapping Study on Approaches for AI-Supported Security Risk Assessment," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021, pp. 755–760.

[3] P. H. Meland, S. Tokas, G. Erdogan, K. Bernsmed, and A. Omerovic, "A Systematic Mapping Study on Cyber Security Indicator Data," *Electronics*, vol. 10, no. 9, 2021.

[4] P. Cheimonidis and K. Rantos, "Dynamic Risk Assessment in Cybersecurity: A Systematic Literature Review," *Future Internet*, vol. 15, no. 10, 2023.

[5] W. S. Admass, Y. Y. Munaye, and A. A. Diro, "Cyber security: State of the art, challenges and future directions," *Cyber Security and Applications*, vol. 2, p. 100031, 2024.

[6] M. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis: The CORAS Approach*. Springer, 2011.

[7] D. Gritzalis, G. Iseppi, A. Mylonas, and V. Stavrou, "Exiting the Risk Assessment Maze: A Meta-Survey," *ACM Comput. Surv.*, vol. 51, no. 1, Jan. 2018.

[8] B. Solhaug and K. Stølen, "The CORAS Language-Why it is designed the way it is," in *11th International Conference on Structural Safety and Reliability (ICOSSAR'13)*. Citeseer, 2013, pp. 3155–3162.

[9] SINTEF Digital, "CORAS Threat Modeler," https://github.com/stverdal/CORAS-The-Explorer/tree/navigator, 2025, accessed: Sep. 20, 2025.

[10] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.

[11] M. H. Heydari, A. Hemmat, E. Naman, and A. Fatemi, "Context Awareness Gate for Retrieval Augmented Generation," in *15th International Conference on Information and Knowledge Technology (IKT)*, 2024, pp. 260–264.

[12] NEMECYS Project, "NEMECYS: NEw MEdical CYbersecurity assessment and design Solutions," https://nemecys.eu/, 2025, accessed: Sep. 21, 2025.

[13] G. Erdogan, R. Halvorsrud, C. Boletsis, S. Tverdal, and J. B. Pickering, "Cybersecurity awareness and capacities of SMEs," in *9th International Conference on Information Systems Security and Privacy (ICISSP)*. SciTePress, 2023, pp. 296–304.

[14] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, T. Yu, Y. Liu, and H. Wang, "Large Language Models for Cyber Security: A Systematic Literature Review," 2025. [Online]. Available: https://arxiv.org/abs/2405.04760

[15] G. Siracusano, D. Sanvito, R. Gonzalez, M. Srinivasan, S. Kamatchi, W. Takahashi, M. Kawakita, T. Kakumaru, and R. Bifulco, "Time for aCTIon: Automated Analysis of Cyber Threat Intelligence in the Wild," 2023. [Online]. Available: https://arxiv.org/abs/2307.10214

[16] E. Aghaei, E. Al-Shaer, W. Shadid, and X. Niu, "Automated CVE Analysis for Threat Prioritization and Impact Prediction," 2023. [Online]. Available: https://arxiv.org/abs/2309.03040

[17] P. Iyenghar, C. Zimmer, and C. Gregorio, "A Feasibility Study on Chain-of-Thought Prompting for LLM-Based OT Cybersecurity Risk Assessment," in *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2025, pp. 1–4.

[18] R. T. Prapty, A. Kundu, and A. Iyengar, "Using Retriever Augmented Large Language Models for Attack Graph Generation," 2024. [Online]. Available: https://arxiv.org/abs/2408.05855

[19] T. Wu, S. Yang, S. Liu, D. Nguyen, S. Jang, and A. Abuadbba, "ThreatModeling-LLM: Automating Threat Modeling using Large Language Models for Banking System," 2025. [Online]. Available: https://arxiv.org/abs/2411.17058

[20] M. S. Salek, M. Chowdhury, M. B. Munir, Y. Cai, M. I. Hasan, J.-M. Tine, L. Khan, and M. Rahman, "A Large Language Model-Supported Threat Modeling Framework for Transportation Cyber-Physical Systems," 2025. [Online]. Available: https://arxiv.org/abs/2506.00831

[21] F. V. Jedrzejewski, D. Fucci, and O. Adamov, "ThreMoLIA: Threat Modeling of Large Language Model-Integrated Applications," 2025. [Online]. Available: https://arxiv.org/abs/2504.18369

[22] S. Roh and T.-S. Kim, "LLM-Based Automated Generation and Tri-Modal Representation of Cyber Attack Scenario," *IEEE Access*, vol. 13, pp. 146 150–146 168, 2025.