Journal of Superintelligence and Computing

CoSci Press

Article

A Case Study on Using Local LLMs for Automated Cybersecurity Analysis

Andrei Brazhuk

Yanka Kupala State University of Grodno, Grodno 230023, Belarus
Corresponding author: brazhuk@grsu.by

Article History: Received 9 July 2026; Revised 20 August 2026; Accepted 2 September 2026; Published 3 September 2026

Abstract: The use of generative AI technologies in architectural threat modeling automation seems to be a promising alternative to 'traditional' formal approaches (e.g., attack-defense trees, domain-specific languages, knowledge graphs, etc.). The main advantage of Large Language Models (LLMs) is that they can work with system and threat descriptions in textual form, skipping the formalization stage required for traditional frameworks. The work considers how by leveraging local LLMs (i.e., running on enterprise hardware) to overcome the data privacy and regulatory compliance risks associated with feeding proprietary data into online AI systems. Threat modeling was performed via the free Gemma 4 model family on a dataset containing 200 descriptions of containerized applications, and results were compared with those from a formal ontology-driven framework. The case study demonstrates that local LLMs can serve as replacements for formal systems when matching and reproducing text-based threat modeling rules. The models with 26 and 31 billion (B) parameters can provide adequate metrics (up to 0.975 precision and 0.981 recall for advanced threat models by the 31B LLM with thinking enabled). However, generative AI cannot guarantee metrics as repeatable as those produced by 'strict' automated cybersecurity analysis systems, and it also requires extra computational power for local use to provide rapid threat modeling.

Keywords: threat modeling automation; generative AI; large language models; ontologies; Ollama; Gemma 4

1. Introduction

The extensive deployment of generative AI and large language models (LLMs) opens up new prospects for various fields, among them cybersecurity. Software engineering tends to establish the "secure by design" principle, which incorporates threat modeling (enumerating potential threats early) to implement security solutions as soon as possible. Architectural flaws are the most costly to fix in terms of time and other resources, whereas mistakes identified during the requirements and design stages are significantly cheaper to correct.

Fast (agile) software development methodologies with automated deployment are widespread, making manual security analysis impractical. Automation plays a critical role in threat modeling, and the use of prospective AI technologies seems to be a good alternative to 'traditional' formal approaches, like attack-defense trees, domain specific languages, ontologies, and knowledge graphs. The main advantage of LLMs is that they can directly take a text description of a system and an informal description of possible threats, and output a threat list. This could accelerate the threat modeling, saving security teams significant time by automating the tedious enumeration process and allowing them to focus immediately on architectural risk analysis.

However, the use of generative AI and LLMs poses two main challenges.

The first challenge involves data privacy and regulatory compliance risks associated with feeding proprietary or personal data into generative AI systems. Since AI services are cloud-based, state and international laws, such as the GDPR (General Data Protection Regulation), impose strict rules regarding data processing, usage, and transfer. Companies also should follow local data protection regulations, which makes compliance even more complex.

The second challenge relates to the need to estimate results produced by AI systems. It is known that LLMs can provide uncertain output (epistemic and aleatoric uncertainty) and incorrect outcomes (hallucinations) due to their probabilistic nature. So, evaluations are required to assess the quality of AI-generated solutions by comparing them with strict formal frameworks.

To keep data privacy and regulatory compliance, local LLMs can be used, which means operating the AI model directly on enterprise hardware rather than using an online service. In addition to data security, it gives us no subscription costs and total ownership of generated results. This has become possible thanks to recent advances in model optimization and the emergence of free LLMs that now support complex AI workloads. For instance, Gemma 4 is a family of open-weight models released in 2026 under a free license. It includes various-sized models, from 2.3 up to 31 billion (B) parameters, optimized for deployment across mobile, edge, and enterprise environments, some of them supports multimodal inputs (text, images, and audio) with a wide context window (256 kilobytes tokens), and a configurable thinking engine.

This work explores the use of local LLMs as an alternative to 'traditional' threat modeling automation, adding the option to work straight with text data and avoiding the formalization stage. The research question (RQ) can be formulated as follows:

RQ: Can local LLMs automate threat modeling from text inputs as effectively as systems based on formal descriptions?

To answer the research question, an open dataset of 200 semantic diagrams [1] has been used in this work. Each diagram of the dataset includes an abstract formal system description based on a container application configuration, and a list of relevant threats associated to diagram items. The threats were obtained from a cloud-based threat model via an ontology-driven framework that automates security analysis [2]. So, dataset [1] has been employed to evaluate local LLMs, and framework [2] is a formal system used for comparison. The threats serve as ground truth to evaluate AI system output.

The primary challenge from a strict system perspective lies in enabling AI to interpret informal indirect threat descriptions, along with direct ones. For instance, a direct description might be like *'User is affected by threat N because the user communicates to Container*' — explicitly identifying 'User' and 'Container' as classes. In contrast, an indirect description might be like *'... because the user communicates to Cloud Application*' which requires AI to infer that Container is a subclass of Cloud Application.

The main contributions of this work are as follows:

1) 200 textual system descriptions have been created from the semantic diagrams [1]. The main advantage of this representation is that every description has a formal list of associated threats from the cloud threat model [2], which can serve as ground truth.

2) A textual description of the cloud threat model [2] has been created. Two AI prompt templates have been added: first - with a particular set of threats related to container applications (direct informal threat patterns); second - extended by generic threats that affect cloud applications (indirect patterns).

3) Based on the textual system descriptions and cloud threat model, evaluation of recent Gemma 4 models (4B, 26B, and 31B) has been performed for AI-based threat modeling, to assess accuracy of local LLMs in matching and reproducing threat modeling rules via classification metrics (precision, recall, and F1-score) and their computational overhead (inference time), including the ability to follow advanced threat patterns, the effectiveness of the thinking mode, and the type of hardware required.

2. Literature Review

Threat modeling and its automation have been considered by the research community for decades [3,4]. 'Traditional' methods for automating architectural security analysis, such as attack-defense trees [5], graph theory [6], domain-specific languages (DSLs) [7], first-order and modal logic [8], knowledge graphs [9], are being extended through generative AI and LLMs [10,11].

This trend began with the advent of public AI services, such as ChatGPT in 2022, which enable a prompt-based approach to generating threat lists from either a given textual system description or source code files [12]. For example, such a 'naive' approach was implemented by the STRIDE GPT tool (https://github.com/mrwadams/stride-gpt).

Research efforts leveraging LLMs and their APIs (both public and local) are primarily focused on studying the features that allow for the enhancement of AI output quality. So, work [13] considers the use of Chain-of-Thought (CoT), prompt tuning (Optimization by PROmpling – OPRO), and Low-Rank Adaptation (LoRA) in order to automate threat modeling for banking systems. Research [14] combined NLP techniques (KeyBERT) and the Llama2 LLM to implement improved Retrieval Augmented Generation (RAG). RAG for automated threat modeling based on ATT&CK, CAPEC, and CVE was also under consideration [15,16].

Although generative AI offers promising results in architectural security analysis, its output needs to be carefully checked [17]. The top objective seems to be to take full power of LLMs and ensure that they provide the same quality as formal systems do. In order to distinguish hype from useful results for the threat modeling field, both massive studies with human participants and creating benchmark systems are required [17,18]. This will be a repetitive process, as new AI technologies arrive, such as multi-agent systems and local LLMs.

3. Data and Methods

Figure 1 provides an overview of the case study.

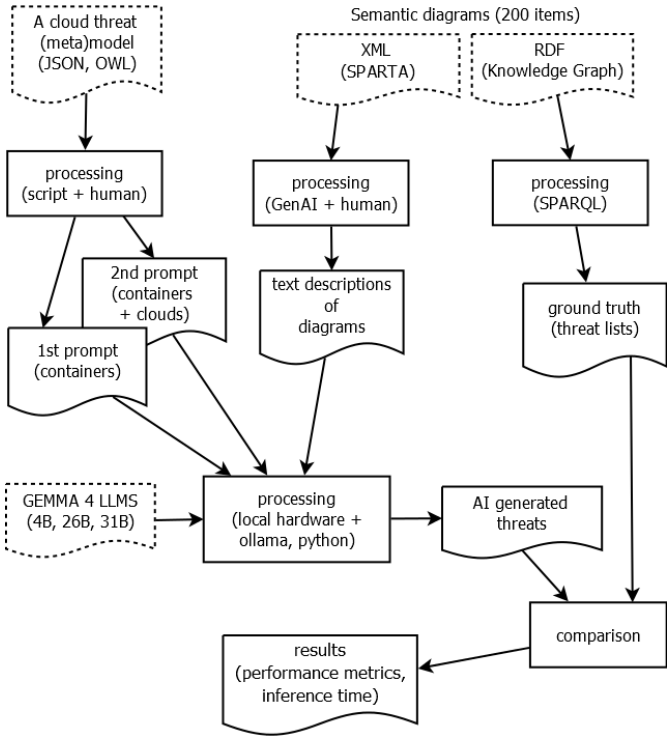


Figure 1. Overview of the case study.

The case study can be divided into two stages:

1) Preparation stage. This includes creating text descriptions for the semantic diagrams [1] (section 3.1), forming two AI prompt templates from the cloud threat (meta)model [2], and enumerating particular threats that can serve as ground truth (section 3.2).

2) Modeling stage. This involves generation of threats for the system descriptions via LLMs based on the prompts, then comparison of AI output with the ground truth to obtain performance metrics and inference time (section 3.3).

All the source data and scripts of the case study have been published on GitHub (https://github.com/nets4geeks/TM-LLM).

3.1. Text Descriptions of Diagrams

Cloud-based (containerized) applications are considered a practical example in this work. Each application may include several components (services) and use some form of storage to save its data. Application structure is represented as a configuration file (docker-compose.yml), called a deployment model.

The dataset of 200 semantic diagrams [1] used in this work contains abstract formal descriptions of applications in the form of Data Flow Diagrams (DFDs). Each description is based on the DFD terminology (External Entity – rectangle in graphical view; Process – circle; Storage – two parallel lines) and also includes container-specific concepts (RemoteUser, Container, ContainerVolume, etc.). Figure 2 shows a graphical example of a DFD.

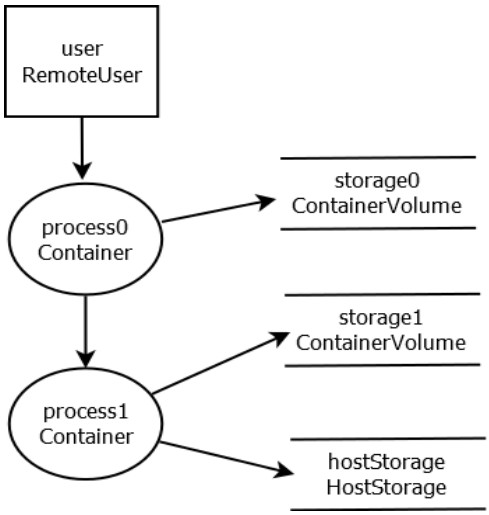


Figure 2. A containerized application (DFD).

A text description of the diagram, shown in Figure 2, might be as follows:

"A client 'user' interacts with the container 'process0'. This service then distributes data to a volume 'storage0' and a second service 'process1'. The second service handles the final stage, pushing data into the storage space 'storage1' and the host storage 'hostStorage'."

So, at this stage, a text description was created for each of the 200 diagrams represented in machine-readable format (XML compatible with the SPARTA threat modeling ecosystem - https://sparta.distrinet-research.be/). Local generative AI (Gemma 4:31b) was instructed to read every diagram and generate five different descriptions for it, after which one was randomly selected. All the descriptions were carefully verified by a human and approximately one-third of the diagrams was corrected (see the 'dataout/text' folder on GitHub).

3.2. Cloud Threat Model

This work is based on a software tool and a set of machine-readable models, united as an Ontology-driven Threat Modeling (OdTM) framework [1,2]. The framework enables a semantic representation (interpretation) of an application configuration as both an ontology in OWL (Web Ontology Language) and a knowledge graph in RDF (Resource Description Framework). An ontological Domain-Specific Threat Model (DSTM) can also be used, which contains formal knowledge about the security aspects of a specific type of computer system. For instance, there exists a DSTM that depicts different aspects of cloud systems, such as generic, organizational, Infrastructure as a Service, and containers. Reasoning and software procedures automatically obtain a threat model based on a Knowledge Base (KB), which combines the semantic interpretation and DSTM.

For this case study, it is crucial that the RDF representation of each of the 200 semantic diagrams includes threat lists formally linked to its items via semantic rules, thereby enabling these threats to be used for evaluating AI-driven security analysis results.

Two prompt templates have been created, which divide a challenge for LLMs into two parts (see 'prompts' on GitHub).

The first template contains (in the 'context' section) direct descriptions of 20 threats related to containers, operating 5 concepts. Each description includes a threat name and a flow pattern ("source – target") with identification of which item is the victim, like:0

"The 'threatEA01_HostStorageUse' is applicable, if Container connects to Host Storage (victim)"

The second prompt template expects AI to recognize inheritance of threats, like:

"All the containers are subclassess of clouds applications, so all the threats related to cloud applications affect containers."

Additionally, to direct threat templates, the second template focuses on indirect descriptions and includes 64 threats (both generic and container-related) which operate on 8 concepts. This could require AI to keep in mind concept inheritance and consider complex relationships between items. An example of a generic threat description is:

"*The 'threatAB01_FailureOfCloudApplication' is applicable, if Remote User (victim) connects to Cloud Application*".

3.3. Local LLMs-based Threat Modeling

A local server used in the case study has been based on an NVIDIA L40S Graphics Processing Unit (GPU). The L40S (https://www.nvidia.com/en-gb/data-center/l40s/) is positioned as an entry-level professional solution. Compared to the NVIDIA RTX 4090, which is a consumer-grade product based on the same Ada Lovelace architecture, the L40S offers greater compute resources - 18,176 CUDA cores (vs. 16,384 on the RTX 4090) and 568 Tensor cores (vs. 512 RTX 4090). The most noticeable distinction is memory capacity: the L40S has 48 GB of GDDR6 memory, whereas the RTX 4090 comes with 24 GB of GDDR6X, which allows the L40S to put an entire LLM within its memory. However, compared to top-tier solutions based on the NVIDIA Hopper (H100) or Ampere (A100) architectures, the L40S is less powerful but comes at a more affordable price.

The rest of the hardware (Table 1) offers sufficient throughput to avoid bottlenecking LLM performance.

Table 1. Hardware configuration.

Item	Amount	Model
GPU	1	NVIDIA L40S
CPU	2	Intel Xeon Gold 6438N
DRAM	16	32 GB DIMM DDR5
Board	1	ASUS Z13PP-D32
SSD	4	6TB INTEL SSDPE2KE064T8

The software part is based on Ubuntu 26.04 with installed Ollama 0.24.0. Ollama (https://ollama.com) is an open-source, cross-platform runtime environment that enables researchers and developers to work with local LLMs via a REST API. Ollama models use the Q4_K_M quantization, a mixed-precision method that quantizes critical tensors at 6 bits and the rest at 4 bits, balancing model compression, inference throughput, and performance retention. Also, Python and shell scripts have been created to implement the threat modeling process (see the 'py' folder of the Github repository).

The Gemma 4 model family has been chosen for several reasons.

Firstly, preliminary research showed that the Gemma 4 set is more effective for the considered tasks than the Llama and Qwen models. The preparatory study consisted of a series of small experiments, primarily conducted on a workstation with a small number of manually created diagram descriptions. Table 2 illustrates such an experiment using 10 diagram descriptions, in which the smaller Gemma variant (4B) demonstrated an advantage over 8B Llama and 9B Qwen, both in terms of standard metrics and resistance to hallucinations.

Table 2. An illustration of preliminary research.

	Model	Precision	Recall	F1	Avg. Time. s	Hallucinations/Total
1	gemma4:e4b	0.726	0.566	0.636	6.14	0/10
2	llama3.1:8b	0.488	0.168	0.250	5.16	2/10
3	qwen3.5:9b	0.645	0.280	0.391	6.13	5/10

Secondly, the Gemma 4 series include models that can be run on either the workstation and server hardware. thereby facilitating reproducibility of experiments across different environments.

And, thirdly, the thinking capability in Gemma 4 is an interesting subject of study. Thinking enables the model to generate internal reasoning traces before formulating a final response, and enhancing performance on multi-step tasks requiring logic, mathematics, and code generation.

The first three columns of Table 3 present the main experiments. The objective was to evaluate the Gemma 4 models with 4B, 26B, and 31B parameters for automated threat modeling on the data set, including 200 text descriptions of container applications. The ground truth for the dataset is known because it relies on the textual representation of the formal threat model.

Table 3. Main experiments.

Exp	Prompt	Model	Thinking	Precision	Recall	F1	Avg. Time, s
1	1st	gemma4:31b	True	0.986	1.000	0.993	151.98
2		gemma4:31b	False	0.981	0.994	0.987	23.05
3		gemma4:26b	True	0.983	0.985	0.984	46.86
4		gemma4:26b	False	0.816	0.782	0.799	4.39
5		gemma4:e4b	True	0.797	0.635	0.707	19.10
6		gemma4:e4b	False	0.510	0.503	0.506	4.50
7	2nd	gemma4:31b	True	0.975	0.981	0.978	382.26
8		gemma4:31b	False	0.937	0.969	0.953	85.83
9		gemma4:26b	True	0.896	0.865	0.880	121.66
10		gemma4:26b	False	0.590	0.582	0.586	11.28
11		gemma4:e4b	True	0.683	0.498	0.576	35.79
12		gemma4:e4b	False	0.502	0.462	0.481	11.09

Three key aspects have been considered.

The first aspect to check is whether local LLMs are able to follow both simple threat patterns (1st prompt), which represents primitive rules, and advanced ones (2nd prompt), which involves inheritance and complex relationships between diagram items.

The second aspect touches on the effectiveness of the thinking mode based on Chain-of-Thought: how much it boosts performance and how it affects computational resources. So, every model has been run twice: with enabled thinking mode and with disabled mode.

The third aspect is whether the considered tasks require workstation or server hardware to run. The 4B model is positioned for use on a workstation, the 26B model as a common enterprise solution, while the 31B model is intended to resolve advanced business challenges. Both the 26B and 31B require server hardware to run.

4. Results and Discussion

Table 3 presents the case study results. 'Precision', 'Recall', and 'F1' are performance metrics calculated in a standard way. 'Avg. Time' represents the average time in seconds for processing one item.

Considering **the simple threat patterns (1st prompt)**, two groups of results have been obtained (Figure 3). The first group is near the highest performance metrics and consists of the 31B (with and without thinking mode) and the 26B (thinking enabled) models. The second group shows around 0,7-0,8 F1 score: the 26B (thinking off) and 4B (thinking on) models.

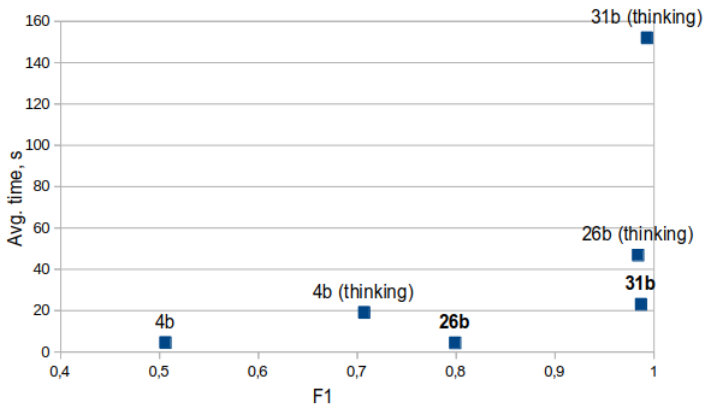


Figure 3. F1 versus average inference time of 1st prompt.

From our perspective, it makes sense to use the 31B model without thinking mode when high precision is required (e.g., for fully automated threat modeling), and the 26B model (thinking off) when it is possible to correct some LLM mistakes. The reason is that these models have the lowest computation time.

Thinking mode is excessive for simple threat patterns: it improves few metrics, but consumes a lot of time (Figure 3).

Considering **the advanced threat patterns (2nd prompt)**, it was found that the 31B model (with thinking both enabled and disabled) and the 26B model (with thinking enabled) provide acceptable results, achieving F1 score ranging from 0,88, while the 31B model with thinking disabled is the fastest (Figure 4).

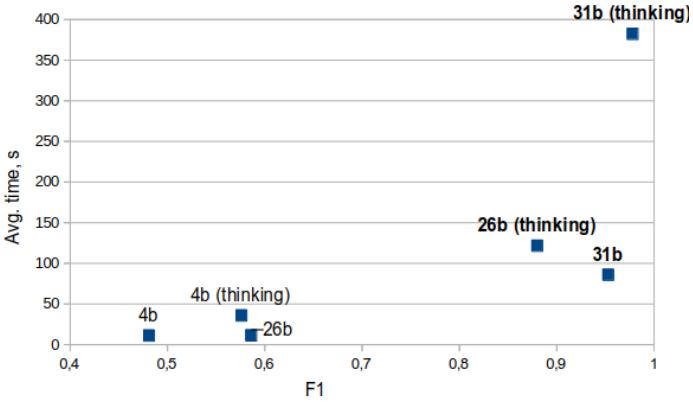


Figure 4. F1 versus average inference time of 2nd prompt.

For the 2nd prompt, the average inference time can be compared with the processing time of the ontology-driven framework on the same hardware (Table 1). So, creating a single dataset item (OWL document, reasoning, TTL and SPARTA files) from a raw Docker Compose file takes 3.5 seconds on average, and fetching ground truth for a single diagram via a SPARQL request takes 0.05 seconds.

In summary, considering the aspects above:

Regarding the first aspect: quantized Gemma 31B and 26B models are able to follow advanced threat patterns with adequate metrics (26B requires thinking mode enabled).

Regarding the second aspect: from our perspective, thinking mode affects more computation time and contributes less to precision and recall. The use of this mode has to be justified.

Regarding the third aspect, while the 4B model can run on client hardware, it is suitable for the simple threat patterns; therefore, server hardware should be used to run the 26B and 31B models for the advanced patterns.

However, computation time remains a challenge for entry-level professional solutions. Waiting minutes to obtain a threat model may be acceptable for semi-automatic use cases, such as manual analysis by a group of experts. For fully automated use cases, a long delay is impractical, especially when integrating threat modeling into a cloud system (automated application deployment) or a CI/CD (Continuous Integration/Continuous Deployment) pipeline.

5. Conclusions

Returning to the research question about the ability of generative AI to work with informal text descriptions of systems and threat models in the same way that current automated frameworks based on formal data do, this study shows that recent local LLMs can provide adequate metrics. The evidence is based on the case study, in which threat modeling was performed on 200 items using Gemma 4 models, and the resulting outputs were compared with those from the formal ontology-driven framework.

The primary practical outcome is that one can begin using models like Gemma 4 with 26- or 31-billion-parameters on enterprise hardware for automated threat modeling based on textual rules. This significantly could shorten the formalization stage, as the models can operate directly on informal text input.

The first issue of generative AI is that it cannot guarantee metrics as repeatable as those produced by 'strict' formal systems. Even the 'smartest' LLMs may produce errors and hallucinations. However, the research community has a lot of methods and great experience for overcoming uncertainty in AI and ML systems.

The second issue lies in the need for extra computational power for the local use. Currently, entry-level server hardware does not provide rapid AI threat modeling: systems based on domain-specific languages, logic, and knowledge graphs seem to be faster.

Funding: This research received no external funding.

Ethical Approval: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: None.

Conflicts of Interest: The author declares no conflicts of interest.

References

[1] Brazhuk, A. Automatic analysis of containerized application deployment models based on ontologies and knowledge graphs. *Int. J. Open Inf. Technol.* **2025**, *13*, 105–111.

[2] Brazhuk, A.I.; Olizarovich, E.V. Ontological analysis in the problems of container applications threat modelling. *Informatics* **2023**, *20*, 69–86. (In Russian)

[3] Xiong, W.; Lagerström, R. Threat modeling–A systematic literature review. *Comput. Secur.* **2019**, *84*, 53–69.

[4] Tran, A.D.; Verreydt, S.; Yskout, K.; Joosen, W. A Multivocal Literature Review on the Effectiveness of Security Threat Modeling. *IEEE Trans. Softw. Eng.* **2026**, *52*, 1352–1370.

[5] Broccia, G.; Ter Beek, M.H.; Lafuente, A.L.; Spoletini, P.; Fantechi, A.; Ferrari, A. Evaluating the understandability and user acceptance of Attack-Defense Trees: Original experiment and replication. *Inf. Softw. Technol.* **2025**, *178*, 107624.

[6] Berger, B.J.; Plump, C. Automatic security-flaw detection-towards a fair evaluation and comparison. *Softw. Syst. Model.* **2025**, *24*, 1763–1796.

[7] Engström, V.; Nebbione, G.; Ekstedt, M. Modeling and Simulating Cyberattacks with the Dynamic Meta Attack Language. *SSRN* **2025**, 5462920 (preprint).

[8] Rouland, Q.; Adi, K.; Timo, O.N.; Logrippo, L. A Formal Approach for Security Pattern Enforcement in Software Architecture. *Comput. Secur.* **2025**, *162*, 104749.

[9] Chiș, A.; Stoica, O.I.; Ghiran, A.M.; Buchmann, R.A. A knowledge graph approach to cyber threat mitigation derived from data flow diagrams. In Proceedings of the 2024 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 16–18 May 2024.

[10] Zhang, J.; Bu, H.; Wen, H.; Liu, Y.; Fei, H.; Xi, R.; et al. When LLMs meet cybersecurity: A systematic literature review. *Cybersecurity* **2025**, *8*, 55.

[11] Yigit, Y.; Buchanan, W.J.; Tehrani, M.G.; Maglaras, L. Review of generative AI methods in cybersecurity. *Internet Things Cyber-Phys. Syst.* **2026**, *5*, 241–261.

[12] Kaheh, M.; Kholgh, D.K.; Kostakos, P. Cyber sentinel: Exploring conversational agents in streamlining security tasks with GPT-4. *arXiv* **2023**, arXiv:2309.16422 (preprint).

[13] Wu, T.; Yang, S.; Liu, S.; Nguyen, D.; Jang, S.; Abuadbba, A. ThreatModeling-LLM: Automating threat modeling using large language models for banking system. *arXiv* **2024**, arXiv:2411.17058 (preprint).

[14] Elsharef, I.; Zeng, Z.; Gu, Z. Facilitating threat modeling by lev-

eraging large language models. In Proceedings of the Workshop on AI Systems with Confidential Computing, San Diego, CA, USA, 26 February 2024.
[15] Salek, M.S.; Chowdhury, M.; Munir, M.B.; Cai, Y.; Hasan, M.I.; Tine, J.M.; et al. A large language model-supported threat modeling framework for transportation cyber-physical systems. *IEEE Access* **2025**, *13*, 163046–163070.
[16] Sammouri, K.; Walden, J. A large language model based threat modeling tool with CAPEC semantic retrieval. *SSRN* **2025**, 5242954 (preprint).
[17] Mbaka, W.B.; Tuma, K. Less is more: usefulness of data flow diagrams and large language models for security threat validation. *Empir. Softw. Eng.* **2026**, *31*, 122.
[18] Bissoli, A.; Mollaeefar, M.; Van Landuyt, D.; Ranise, S. Benchmarking the effectiveness of multi-agent LLMs in collaborative privacy threat modeling with LINDDUN GO. *J. Inf. Secur. Appl.* **2026**, *100*, 104489.



Copyright: © 2026 by the author(s). *Journal of Superintelligence and Computing* supported by Wuchang University of Technology and published by CoSci Press. This is an open access publication under the Creative Commons Attribution 4.0 International License (CC BY 4.0).