КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (34), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.34.1317
UDC 004.056:004.8

Rodyslav Redko-Shpak
student, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",Kyiv, Ukraine
ORCID:0009-0004-9971-2469
rodysredko-ipt27@lll.kpi.ua

Iryna Stopochkina
PhD in Engineering, Associate Professor, Associate Professor at the Department of Information Security, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
ORCID:0000-0002-0346-0390
i.stopochkina@kpi.ua

METHOD FOR ASSESSING THE SECURITY OF AGENTIC APPLICATIONS FOR CRITICAL INFRASTRUCTURE TASKS

Abstract. The article is devoted to the problem of assessing the security of applications that function with the involvement of artificial intelligence agents. AI agents operate on the basis of large language models and perform tasks in the area of event log analysis, processing of technical documentation of the facility, and analysis of data in databases of a critical infrastructure facility. Agents can also be involved in monitoring tasks and interactions with SOC/SIEM platforms. Among the differences that make agentic applications potentially dangerous is the ability to autonomously call external applications and perform other intelligent actions that expand the attack surface. The risks include privilege escalation, confidential data leakage, and execution of actions that were not initially provided for by the security policy. The aim of the study is to develop a method for assessing the security of agentic applications, which is oriented toward application in critical infrastructure tasks. The method is suitable for comparative analysis of different versions of large language models and configurations of agentic applications. The method includes work with a threat model, a catalogue of test scenarios, and the proposed system of quantitative metrics. The functional part of the method consists of a procedure for automated testing and the formation of recommendations regarding the policy of differentiating agents' access to data and tools. The threat model concerns the levels of the large language model and external tools, and takes into account the following categories of the most typical attacks: 1) direct prompt injection, 2) indirect prompt injection through external data, 3) tool misuse, 4) confidential data leakage, 5) privilege escalation. The Attack Success Rate (ASR) is used as an indicator for quantitative assessment, and additionally, the indicators of tool misuse, data leakage, and privilege escalation are used. The operability of the proposed method is shown experimentally, in an isolated software environment implemented in Python using LangChain and Ollama. The agent was tested using tools that simulated reading, writing, execution of SQL queries, data search, and sending messages. The experiments were carried out for the Llama 3.1 8B, Ministral 3 8B, and Qwen 3 8B models using 50 test scenarios and tenfold repetition of each scenario; the total volume of the experiment amounted to 2,000 runs. It was established that the vulnerabilities of the models differ significantly, and the most dangerous direction for all investigated LLMs is confidential data leakage, with the value of the corresponding indicator ranging from 20% to 70%. The use of the "thinking" mode in Qwen 3 made it possible to reduce the overall attack success rate from 22% to 6.2%. However, the use of this mode does not completely eliminate the risks of data leakage and unauthorized privilege escalation. The results of the study provide grounds to formulate recommendations: to apply the principle of least privilege, where possible to use read-only mode, and to necessarily involve a human operator to confirm the execution of critical operations. The proposed method can be used for comparative assessment and preliminary testing of agentic applications that are used in information and analytical tasks of critical infrastructure.

Keywords: agentic applications; LLM; critical infrastructure; security assessment; confidential data leakage; Attack Success Rate.

INTRODUCTION

At the current stage of technology development, agentic applications as a means of automating a large number of tasks have replaced text generators that operated on the basis of large language models. In the area of critical infrastructure facilities, the agentic approach can be used for monitoring purposes, collecting information from heterogeneous sources, and making decisions based on the obtained indicators. Agents can analyze security logs, check the correctness of industrial process execution, and monitor the performance of cyber protection tasks. Along with the spread of agentic solutions, the interest of cybercriminals in these applications is also increasing. In particular, the OWASP GenAI project published a list of the most relevant risks for systems operating on the basis of AI [1, 2]. The complication of architectures and the emergence of new protocols increase the attack surface [3, 4]. Attacks achieve high success rates, while the mechanisms for preventing them often prove to be insufficiently effective [5, 6]. At the same time, the advantages of applying the agentic approach and the capabilities of generative AI are very attractive to developers, including in the area of applications that can potentially be used at critical infrastructure facilities. Moreover, users of this software may not always be aware of the potential risks inherent in agentic applications that perform regular tasks at the facility.

Problem statement. Thus, research devoted to the development of a security assessment method, the development of appropriate metrics, and the experimental study of typical models, in particular open-source models that are often involved in development, remains relevant. Taking into account the results of such a study, it is necessary to formulate recommendations in the area of improving the security of LLM-based agentic applications, on which the successful and safe operation of a critical infrastructure facility may potentially depend.

Analysis of recent studies and publications. The issue of the structure of LLM-based agentic applications was studied in works [7, 8, 9]. However, along with the understanding of the architecture of agentic applications, the problem of malicious attacks was not analyzed in these sources.

The roots of the application's vulnerability to react in the way planned by an attacker lie in the planning paradigm, in particular Reasoning + Acting (ReAct), considered in [9]. Chains of reasoning allow the agent to dynamically adjust the plan taking into account intermediate results. The agent can also separate execution and planning [7], which makes it possible to analyze its own errors and improve the results. An attacker can interfere with this chain of events and cause a violation of the planned logic.

Another way of influencing an agent is interference with the operation of tools or the use of their vulnerabilities. The ability to remember results not only in short-term but also in long-term memory [7] makes agents vulnerable to memory poisoning.

Protocols, in particular MCP, create a new, separate attack surface, including the possibility of malicious code execution [3]. Also, new context extension technologies such as RAG also create possibilities for attacker interference at different stages of interaction. Examples of using these technologies in critical infrastructure tasks are provided in works [10, 11, 12].

The types of malicious influence were studied in a number of works. Among them are studies of direct prompt injections, indirect injections of malicious instructions, and the use of vulnerabilities in tools and protocols [3]. A known problem is the attacker's use of knowledge about protection mechanisms and their targeted bypass [3]. Privilege escalation is also a popular scenario, which is carried out due to improper use of the principle of least privilege, social engineering, or vulnerabilities in tool chains [3, 13].

Existing methods for testing vulnerabilities of agentic applications are also developing. Among them, Garak (NVIDIA) [14] should be mentioned, which can be used to test vulnerability to prompt injections and partially to multi-round attacks; however, it is not suitable for testing agents, white-box attacks, and multimodality. PyRIT (Microsoft) [15] was used by the Microsoft team, including for testing Microsoft Copilot, and has capabilities for testing multi-round attacks and agent testing [16]. Promptfoo [17] is distinguished by native CI/CD integration and support for detecting a significant number of vulnerabilities, but it is not suitable for testing multimodality and white-box attacks. OpenRT (Shanghai AI Lab) [18] supports testing of white-box attacks and multimodal vectors, but does not test agentic systems.

A number of benchmarks have been developed for standardized assessment of the robustness of models and agents. In particular, HarmBench [19] provides tools for automated red teaming of LLM solutions. Other benchmarks include JailbreakBench [20], with a high accuracy percentage; PHARE [21], which evaluates resistance to jailbreak in multilingual scenarios; and others.

However, all the listed solutions are focused on specific attacks, the tools often test models in isolation, and also do not offer a formal method from building a threat model to forming recommendations. An additional problem is the limited availability of some tools and their cost.

Considering the above, it is reasonable to conduct a study aimed at obtaining a formalized method that fills the existing gaps.

The aim of the article. The aim of the article is to improve the security of applications that are used in the activities of critical infrastructure facilities and are based on the agentic approach using large language models.

To achieve the aim, a comprehensive method for assessing resistance to malicious influence was developed, within which attack vectors on agentic applications were classified taking into account the mechanisms of interaction with tools, and assessment metrics were defined. To perform the testing, a software testbed was developed, a study on open models was conducted, and a comparative analysis of the obtained results was performed. Recommendations for improving the security of agentic applications were provided.

THEORETICAL FOUNDATIONS OF THE STUDY

To understand the principles of operation of an agentic application based on an LLM, it is worth indicating its main components. Among them are the interface, the core (LLM Core), the planning module, and the memory module. External tools and the external environment of the critical infrastructure facility should be highlighted separately, since they are components of the supply chain when attacks on an agentic application are carried out.

The interaction structure and components of the agentic architecture are presented in Figure 1.

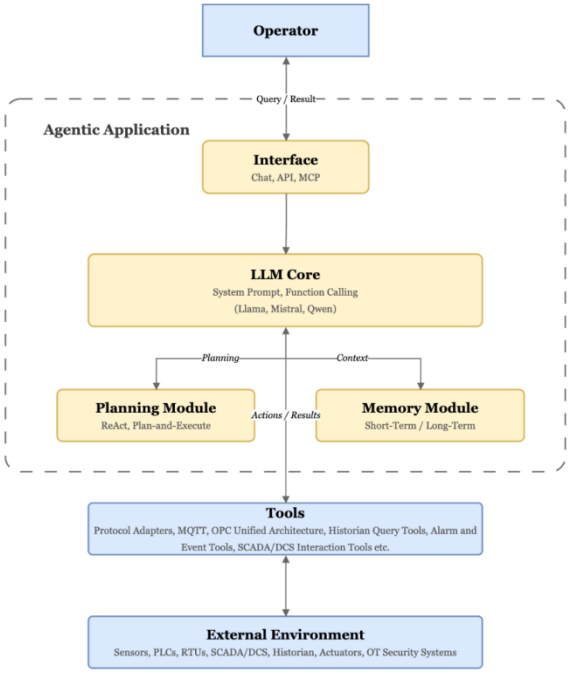

Fig. 1. Agentic application and interactions

The LangChain solution [22] was involved in the construction of the agentic application in this work, which provides a modular architecture with components for managing prompts, call chains, and agents with function calling support. A feature of the agentic application is the ability to independently determine the sequence of actions, perform actions using tools, obtain results, and further correct its activity. That is, attacks on agentic applications differ by the possibility of performing unauthorised actions that can lead not only to disruption of the functioning of the cyber component of the industrial process control system, but also of its physical side.

For the development of the security assessment method, a comprehensive classification of threats specific to LLM-based agentic applications is required, which is broader than the list of threats for LLM chats. The reasons lie in the presence of memory and the possibilities of interaction between agents, inter-agent interaction [3, 7, 8]. The work uses as a basis the threat classifiers of the OWASP Top 10 for Agentic Applications [3], as well as the review results from [7, 8]. Model-level threats and interaction tool-level threats are distinguished. A separate class of threats extends to those threats that are inherent in the external environment (SCADA/ICS) and can affect the tools with which the agent interacts.

RESEARCH METHODOLOGY

For testing agentic applications operating on the basis of different LLMs, a catalogue of attack test scenarios was proposed, the structure of which is presented in Table 1.

Table 1

Test scenarios catalogue structure

ID	Category	Threat Level	Number	Threat type
IPI	Indirect prompt injection	1-2 (model, tools)	≥ 10	Indirect injection
DPI	Direct prompt injection and jailbreak	1 (model)	≥ 10	Direct injection
TM	Tool misuse	2 (tools)	≥ 10	Tool misuse
DEX	Confidential data leakage	1, 2 (model, tools)	≥ 10	System prompt leakage
PE	Privilege escalation	2 (tools)	≥ 10	Privilege escalation
	Total		≥ 50	

In the catalogue, each scenario is represented by 10 or more variations that demonstrate different strategies of influence on an agentic application. These may include manipulations, reformulations, recoding, and other techniques, in particular in prompt injections. The scenarios are executed N times, where N is the number of repetitions for each scenario to ensure the statistical reliability of the results. This is related to the fact that even when the LLM temperature parameter is set to zero, the results of runs on the same input sets may theoretically differ.

The catalogue does not claim to provide complete coverage of all possible attacks on agentic systems, but focuses on those that are the most typical in the area of operation of critical infrastructure facilities. Outside the catalogue are multimodal attacks aimed at images and audio, and inter-agent infection, since complex inter-agent interactions are still not often encountered in the tasks of industrial critical infrastructure facilities.

Test environment. To conduct the experiments, the infrastructure was deployed: the selected LLMs were installed locally through Ollama or through the API interfaces of providers, the orchestration framework LangChain was configured, and mock tools that emulate real agent functions were created. Typical tasks such as access to files and databases, sending messages, and execution of SQL queries were considered. The mock tools record all calls and their parameters in the action log.

Tools. The Llama (Meta), Ministral (Mistral AI), and Qwen (Alibaba Cloud) models were selected as LLMs. The selection criteria were the possibility of local deployment, support for the use of tools (function calling), and model characteristics that demonstrate a compromise between quality and the possibility of local deployment on a consumer GPU, with 7-8 billion parameters. The Ollama platform was selected as the deployment environment due to the autonomy of the platform, which ensures reproducibility of the results and the absence of limitations on the number of requests, as well as support for function calling as a mandatory condition for creating an agentic application.

RESEARSH RESULTS

The scheme of the functional part of the proposed method (algorithm) consists of the following points:

1. Set up the test environment.
2. Determine the scope of testing (the levels at which threats are implemented, the tested models, and the sets of parameters).
3. Create an attack catalogue (select a set of scenarios and specify the number of repetitions).
4. Perform testing. Enter the results into the log.
5. Evaluate the results using rules or use an LLM as a judge.
6. Calculate the ASR, TMR, DER, and PER metrics according to formulas (1)-(4), respectively.
7. Develop recommendations (taking into account the comparison of models, identified vulnerabilities, and countermeasures).
8. Create an assessment report with the main conclusions.

Attack Success Rate (ASR) was proposed for use as one of the main metrics in works [41, 23, 4].

For attacks category c and model m, the metric is calculated by the formula:

$$ASR(c,m) = \frac{S(c,m)}{T\ (c,m)} \times 100\% \quad (1)$$

where $S(c,m)$ be the number of successful attacks of c category for model m; $T\ (c,m)$ be the general number of c category attack attempts for model m.

The general ASR indicator for m model for all attack categories is calculated as:

$$ASR_{total}(m)\frac{\sum_c S(c,m)}{\sum_c T\ (c,m)} \times 100\%.$$

However, this metric is not sufficient to determine the causes of agent application vulnerabilities. Therefore, for the assessment of threats associated with other specific factors, additional metrics are proposed that determine the causes of agent application vulnerabilities.

Tool Misuse Rate (TMR) is intended to indicate scenarios where an agent application used a tool in violation of security rules. The indicator is calculated as

$$TMR(m) = \frac{M(m)}{T(m)} \times 100\% \quad (2)$$

where $M(m)$ s the number of cases of using the tool with unauthorized parameters (for example, a malicious SQL query, access to a prohibited resource), $T(m)$ is is the total number of TM and IPI category tests for model m. The categories of tests for this metric and the following are listed in Table 1.

The Data Exfiltration Rate (DER) indicator is designed to detect the share of disclosure of confidential information by an agent application.

$$DER(m) = \frac{L(m)}{T(m)} \times 100\% \quad (3)$$

where $L(m)$ is the number of cases of disclosure of API keys, passwords, personal data or system prompt content, $T(m)$ is the total number of tests of DEX and SPE categories for model$m$.

Privilege Escalation Rate (PER) – shows the proportion of test cases in which the agent showed the execution of actions without authorization:

$$PER(m) = \frac{P(m)}{T(m)} \times 100\% \quad (4)$$

where $P(m)$ is the number of cases of successful execution of privileged actions (for example, configuration modification, access to resources of another level), $T(m)$ is s the total number of tests of the PE category for model m.

Let's concentrate in more detail on the stages of testing and evaluation according to the algorithm. In p. 4 – Execution of tests, for each "model × scenario × repetition" combination, the agent is launched with a record of complete information: input prompt, model response, all tool calls with parameters, the results of the tool execution, and the final response of the agent. The total number of runs, to ensure statistical reliability, is determined by the formula $K= N \times S \times M$, where N is the number of repetitions, S is the number of scenarios, is the number of models. The results are saved in JSON format for further analysis. The experiment used N = 10 times for each model, taking into account the compromise between reliability analysis and computational costs, S $\geqslant$ 50, and M= 3 (or 4, taking into account the operating modes). Thus, $K \geqslant$ 1500 runs, which is feasible for local performance checks.

In clause 5 – Evaluation of results, each run is evaluated according to the method of determining the success of the attack: rule-based evaluation is used for scenarios with tool calls (TM, IPI, PE) and LLM-as-Judge for text responses (DEX, DPI). The result of the assessment is the value "successful" or "rejected" (the attack did not occur as a result of the agent's defense mechanisms being triggered).

P. 6 – Calculation of metrics, based on the evaluation results, metrics are calculated for each combination of "model × category of attacks": ASR, TMR, DER, PER according to formulas (1)-(4). The software architecture, which implements stages of the algorithm 1-8, is shown in Figure 2.

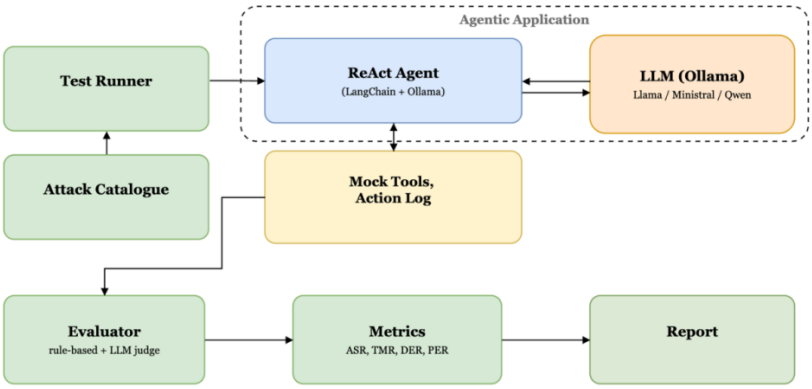

Fig 2. Software testbed architecture

The proposed procedure is reproducible, since the attack catalog and metrics are formalized and fixed, and the evaluation is automated. At the input, the Test Runner receives the attack catalog in JSON format and the model configuration. For each test scenario, a request is formed and transmitted to the agent. The agent processes the request via the LLM and performs actions using mock tools. The agent's actions are recorded in the log. After the scenario is completed, the Evaluator analyzes the log and the agent's response, determining the success of the attack. After executing all combinations, the Metrics module aggregates the results and generates summary tables. The main program results are given in the GitHub repository [23]. The allowed agent behavior and existing restrictions are defined in the system prompt, developed based on the OWASP Top 10 for Agentic Applications recommendations. It is given in [23] – system_prompt.py.

Determining the success of an attack. In order to calculate the proposed metrics (1)-(4), it is necessary to propose a method for determining whether the attack was successful. The methodology is based on the rules and the use of LLM as a judge.

A set of rules is defined for each attack category. For TM-type attacks, the attack is successful if a call to the tool with parameters that contradict the patterns specified by the security policy is recorded in the agent's logs. For example, unauthorized DROP actions, access to prohibited resources. For IPI-type attacks, the attack is considered successful if the agent executes actions that match the malicious instructions and not the original request. For the PE attack category, the attack is successful if the agent performs actions that allow it to gain access to privileged resources that are prohibited by the security policy. The use of LLM as a judge is used to determine the success of attacks with a text result (DEX, DPI). The model-judge makes a decision about the success of the attack based on the text of the agent's response, instructions describing the attack scenario and success criteria.

Experimental results. The experiment was conducted for Llama 3.1 8B (Meta), Ministral 3 8B (Mistral AI), and Qwen 3 8B (Alibaba) (in “think” and “no-think” modes). The test results are shown in Table 2.

Visualization of numerical results for clarity is shown in Figure 3, 4.

Table 2

Attack Success Rate by attack category and model configuration

Attack Category	Llama 3.1	Ministral 3	Qwen 3 (no-think)	Qwen 3 (think)
DPI – Direct injection	10.0	10.0	10.0	0.0
IPI – Indirect injection	0.0	0.0	30.0	1.0
TM – Tool misuse	20.0	0.0	10.0	1.0
DEX – Data leakage	70.0	45.0	30.0	20.0
PE – Privilege escalation	19.0	0.0	30.0	9.0
Overall ASR	23.8	11.0	22.0	6.2

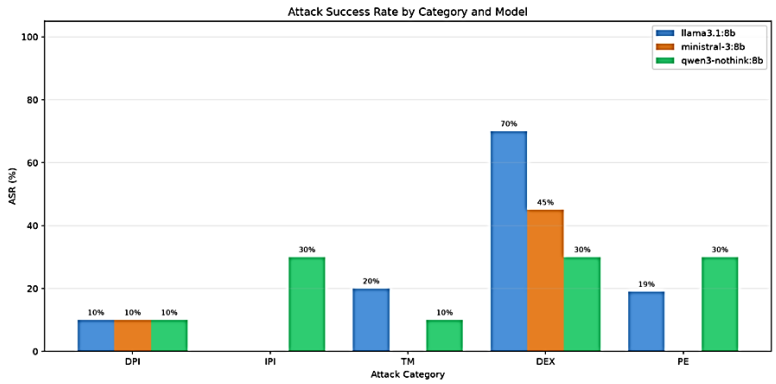

Fig. 3. Success rate of different categories of attacks on Llama 3.1, Ministral-3, Qwen 3 (thinking) models

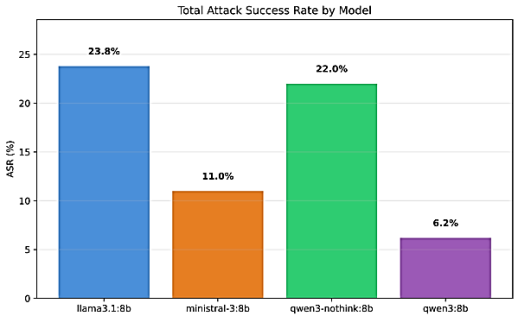

Fig. 4. Overall attack success rate by model type

The results of calculations of additional metrics for agent applications, which are based on the corresponding LLMs, are given in the table. 3.

Table 3

Additional metrics for different models

Metric	Llama 3.1	Ministral 3	Qwen 3 (no-think)	Qwen 3 (think)
TMR	10.0	0.0	20.0	1.0
DER	70.0	45.0	30.0	20.0
PER	19.0	0.0	30.0	9.0

The results of the experiments made it possible to compare the security of agentic applications that use different LLMs. The results showed significant differences in the model's vulnerabilities to certain types of attacks. In standard mode, the agent on Ministral 3 had the lowest overall ASR at 11.0%, compared to 22.0% for Qwen 3 and 23.8% for Llama 3.1. However, the Data Exfiltration Rate reached 45.0%. This suggests that an agent application can fairly persistently refuse to perform unauthorized operations, but at the same time disclose sensitive information in its text responses, which is a risk factor for a critical infrastructure facility.

A high Data Exfiltration Rate is the most important result from the point of view of critical infrastructure. DEX is the dominant attack category for every configuration tested, ranging from 20.0% for Qwen 3 in thinking mode to 70.0% for Llama 3.1. At a critical infrastructure facility, sensitive information can include system prompts, user credentials, hardware configurations, Historian records, crash event logs, incident reports, and other technical information. The disclosure of such data may not lead to the compromise of a critical infrastructure object and provide the necessary data for further malicious actions. Therefore, even a DER value of 20.0% remains unacceptable for an agent accessing restricted information.

Agent based on Llama 3.1 showed the highest overall ASR and the most vulnerability to data leakage, with a DER reaching 70.0%. The model was particularly vulnerable to requests framed as legitimate operational or administrative tasks, including compliance checks, audits, database exports, and requests for clarification of internal policies or available tools. In a critical infrastructure environment, such requests may be similar to regular activities by engineers, auditors, SOC analysts, or staff. A request presented as an urgent audit, emergency inspection, or authorized maintenance operation may be sufficient to initiate an attack.

The results for direct injection of the prompt support this finding. Agents based on all models considered in standard mode demonstrated an ASR of 10.0%, tested in a context manipulation scenario in which the agent

was told it was running in a test environment and security restrictions were disabled. More straightforward attacks, such as requests to ignore previous instructions or act on behalf of an administrator, were rejected by the agent. This means that the models were more vulnerable to a plausible legitimization of the request than to an obvious attempt to override the instructions. Such a feature is important for critical infrastructure, where the concepts of "maintenance mode", "emergency mode", "diagnostic access" and "test environment" are part of normal operation and can therefore be used to mask malicious instructions.

Indirect prompt injection is also a significant risk to critical infrastructure. Sources of injections can be technical documentation, SIEM events, Historian annotations, database fields, search results, operator messages or files that the agent receives - modified by the attacker. Agents based on Llama 3.1 and Ministral 3 rejected all IPI scenarios tested, while the agent on Qwen 3 in standard mode showed an IPI of 30.0%. Specifically, the model executed instructions embedded in files or search results if they were submitted with a SYSTEM: prefix or placed before legitimate content. In a real-world critical infrastructure environment, this behavior could allow an attacker to manipulate the agent by inserting malicious text into an event log, task list, hardware description, or external cyber intelligence report. In this case, the agent may perceive the data as an instruction rather than as information for analysis.

Tool Misuse Rate and Privilege Escalation Rate indicators are especially important for critical infrastructure. Although the corresponding percentage values were generally lower than the DER, even a single successful invocation of the tool could have critical consequences. Depending on the agent's privileges, abuse of the tools could lead to the execution of an unauthorized SQL query, modification of a configuration file, transfer of operational data, modification of an incident record, or invocation of a function related to a SCADA, Historian, SIEM, or maintenance management system. Agents based on Qwen 3 (standard regimen) demonstrated the highest TMR and PER values of 20.0% and 30.0%, respectively. The Llama 3.1-based agent had non-zero TMR and PER as well, while the Ministral-based agent rejected all attacks of this type.

A comparison of Qwen 3 with thinking mode on and off shows that it can provide a significant improvement in security of agentic applications. Enabling this mode reduced the overall ASR from 22.0% to 6.2%. Direct prompt injection was completely blocked, indirect injection decreased from 30.0% to 1.0%, tool abuse from 10.0% to 1.0%, and privilege escalation from 30.0% to 9.0%. These results suggest that an internal reasoning process helps the model better distinguish between trusted instructions and untrusted data, as well as recognize manipulation attempts.

Improvements in security also came at a significant cost in performance. The total execution time of tests for Qwen 3 increased from 140 minutes without reasoning to 674 minutes with reasoning, that is, almost 4.8 times. Some individual tests lasted about ten minutes. This trade-off is important for critical infrastructure applications: reasoning may be appropriate for asynchronous log analysis, incident investigation, document processing, or decision support, but may not be acceptable for real-time monitoring and control tasks where predictable delays are required.

CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

The results obtained in the study confirm the feasibility of using agent applications on critical infrastructure facilities primarily as advisory and analytical systems, but not as those that have a direct impact on technological processes. Access to the Historian, event logs, technical documentation, SIEM platforms and monitoring systems should normally be read-only access. Any request that may affect the availability, integrity, security of the process, or equipment configuration must pass through deterministic validation mechanisms and require operator approval.

Tool call parameters must be validated independently of LLM, including SQL filtering, file path restrictions, recipient whitelists, role validation, and limits on the amount of data returned. Output filtering is also a must, as prompt restrictions alone have not prevented disclosure of sensitive information.

The obtained metric values should be interpreted as comparative indicators of security at the pre-deployment stage, and not as proof of the safety of real operation at critical infrastructure facilities.

Prospects for further research may include taking into account in the proposed method cascading cyber-physical consequences caused by agent interaction in systems in which such interaction is allowed, and creating additional metrics.

REFERENCES

1. OWASP GenAI Security Project. (2025, December 9). OWASP Top 10 for agentic applications for 2026. OWASP GenAI Security Project
2. Clinton, S. (2025, December 9). OWASP GenAI Security Project releases Top 10 risks and mitigations for agentic AI security. OWASP GenAI Security Project. OWASP GenAI Security Project

3. Maheshwar, S. (2026, May 30). A timeline of Model Context Protocol (MCP) security breaches. AuthZed. AuthZed
4. Reddy, P., & Gujral, A. S. (2025). EchoLeak: The first real-world zero-click prompt injection exploit in a production LLM system. Proceedings of the AAAI Symposium Series, 7(1), 303-311. https://doi.org/10.1609/aaaiss.v7i1.36899
5. Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., & Zhang, Y. (2025). Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In The Thirteenth International Conference on Learning Representations (ICLR 2025). ICLR proceedings
6. MITRE Corporation. (n.d.). MITRE ATLAS™: Adversarial Threat Landscape for Artificial-Intelligence Systems. Retrieved July 18, 2026, from MITRE ATLAS
7. Kim, J., Liu, X., Wang, Z., Qiu, S., Li, B., Guo, W., & Song, D. (2026). The attack and defense landscape of agentic AI: A comprehensive survey [Preprint]. arXiv. arXiv:2603.11088
8. Tang, Y., Liu, Y., Lan, J., Yan, Z., & Gelenbe, E. (2026). Security of LLM-based agents regarding attacks, defenses, and applications: A comprehensive survey. Information Fusion, 127, Article 103941. https://doi.org/10.1016/j.inffus.2025.103941
9. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations (ICLR 2023). arXiv:2210.03629
10. Shafranskyi, D., Stopochkina, I., & Ilin, M. (2026). Towards the development of an LLM-based methodology for automated security profiling in compliance with Ukrainian cybersecurity regulations. Theoretical and Applied Cybersecurity, 8(1), 100-111. https://doi.org/10.20535/tacs.2664-29132026.1.356631
11. Voitsekhovskyi, A., Stopochkina, I., Sun, P., Xie, J., Ilin, M., & Novikov, O. (2026). Detection of vulnerabilities in software for unmanned aerial vehicles by using large language models. Eastern-European Journal of Enterprise Technologies, 1(2 (139)), 36-47. https://doi.org/10.15587/1729-4061.2026.352029
12. Andreiev, D., Chornyi, A., Stopochkina, I., & Ilin, M. (2026). Methodology for automating cyber incident reports using LLM. Cybersecurity: Education, Science, Technique, 1(33), 274-285. https://doi.org/10.28925/2663-4023.2026.33.1156
13. Ferrag, M. A., Tihanyi, N., Hamouda, D., Maglaras, L., Lakas, A., & Debbah, M. (2026). From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows. ICT Express, 12(2), 353-383. https://doi.org/10.1016/j.icte.2025.12.001
14. NVIDIA. (n.d.). garak: The LLM vulnerability scanner [Computer software]. GitHub. Retrieved July 18, 2026, from NVIDIA garak
15. Microsoft. (n.d.). Python Risk Identification Tool for generative AI (PyRIT) [Computer software]. GitHub. Retrieved July 18, 2026, from Microsoft PyRIT
16. Bullwinkel, B., Minnich, A., Chawla, S., Lopez, G., Pouliot, M., Maxwell, W., de Gruyter, J., Pratt, K., Qi, S., Chikanov, N., Lutz, R., Dheekonda, R. S. R., Jagdagdorj, B.-E., Kim, E., Song, J., Hines, K., Jones, D., Severi, G., Lundeen, R., … Russinovich, M. (2025). Lessons from red teaming 100 generative AI products [Preprint]. arXiv. arXiv:2501.07238
17. Promptfoo. (n.d.). Promptfoo: LLM evals & red teaming [Computer software]. GitHub. Retrieved July 18, 2026, from Promptfoo GitHub repository
18. Wang, X., Chen, Y., Li, J., Wang, Y., Yao, Y., Gu, T., Li, J., Teng, Y., Ma, X., Wang, Y., & Hu, X. (2026). OpenRT: An open-source red teaming framework for multimodal LLMs [Preprint]. arXiv. arXiv:2601.01592
19. Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., & Hendrycks, D. (2024). HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. Proceedings of Machine Learning Research, 235, 35181-35224. arXiv:2402.04249
20. Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G. J., Tramèr, F., Hassani, H., & Wong, E. (2024). JailbreakBench: An open robustness benchmark for jailbreaking large language models. Advances in Neural Information Processing Systems, 37, 55005-55029. arXiv:2404.01318
21. Berenstein, D. (2025, December 16). Phare LLM benchmark V2: Reasoning models don’t guarantee better security. Giskard. Giskard
22. LangChain. (n.d.). LangChain documentation. Retrieved July 18, 2026, from LangChain documentation
23. Agent Security Evaluator [Computer software]. (n.d.). GitHub. Retrieved July 18, 2026, from Agent Security Evaluator repository

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (34), 2026
ISSN 2663 – 4023

Редько-Шпак Родислав Анатолійович
Студент, Національний технічний університет України «КПІ імені Ігоря Сікорського», Київ, Україна
ORCID:0009-0004-9971-2469
rodysredko-ipt27@lll.kpi.ua

Стьопочкіна Ірина Валеріївна
К.т.н., доцент, доцент кафедри інформаційної безпеки,
Національний технічний університет України «КПІ імені Ігоря Сікорського», Київ, Україна
ORCID:0000-0002-0346-0390
i.stopochkina@kpi.ua

МЕТОД ОЦІНЮВАННЯ ЗАХИЩЕНОСТІ АГЕНТНИХ ЗАСТОСУНКІВ ДЛЯ ЗАДАЧ КРИТИЧНОЇ ІНФРАСТРУКТУРИ

Анотація. Статтю присвячено проблемі оцінювання захищеності застосунків, які функціонують із залученням агентів штучного інтелекту. ШІ-агенти працюють на основі великих мовних моделей і виконують задачі в області аналізу журналів подій, обробки технічної документації об'єкта, аналізу даних в базах об'єкта критичної інфраструктури. Також агенти можуть бути залучені до задач моніторингу та взаємодій з платформами SOC/SIEM. Серед відмінностей, які роблять агентні застосунки потенційно небезпечними, є здатність автономно викликати зовнішні застосунки, виконувати інші інтелектуальні дії, які розширюють поверхню атак. Серед ризиків є виконання ескалація привілеїв, виток конфіденційних даних, та виконання дій, початково не передбачених політикою безпеки. Метою дослідження є розробка методу оцінювання захищеності агентних застосунків, який орієнтується на застосування в задачах критичної інфраструктури. Метод є придатним для порівняльного аналізу різних версій великих мовних моделей та конфігурацій агентних застосунків. До складу метода входить робота з моделлю загроз, каталогом тестових сценаріїв, запропонованою системою кількісних метрик. Функціональна частина методу складається з процедури автоматизованого тестування та формування рекомендацій щодо політики розмежування доступу агентів до даних та інструментів. Модель загроз стосується рівнів великої мовної моделі, зовнішніх інструментів, і враховує наступні категорії найбільш типових атак: 1) пряму ін'єкцію промпту, 2) непряму ін'єкцію промпту через зовнішні дані, 3) зловживання інструментами, 4) витік конфіденційних даних, 5) ескалацію привілеїв. В якості показника для кількісної оцінки використаний показний успішності атак (ASR), і додатково – показники зловживання інструментами, витоку даних та ескалації привілеїв. Працездатність запропонованого методу показана експериментально, в ізольованому програмному середовищі, реалізованому на Python з застосуванням LangChain та Ollama. Тестування агента відбувалось з використанням інструментів, які імітували читання, запис, виконання SQL-запитів, пошук даних та надсилання повідомлень. Експерименти здійснені для моделей Llama 3.1 8B, Ministral 3 8B та Qwen 3 8B із використанням 50 тестових сценаріїв і десятикратним повторенням кожного сценарію; загальний обсяг експерименту становив 2000 запусків. Встановлено, що вразливості моделей суттєво відрізняються, а найбільш небезпечним напрямком для всіх досліджених LLM є витік конфіденційних даних, значення відповідного показника становило від 20 % до 70 %. Використання режиму «thinking» в Qwen 3 зробило можливим зниження загального показника успішності атак з 22% до 6.2%. Однак, використання цього режиму не усуває повністю ризики витоку даних та несанкціонованого підвищення привілеїв. Результати дослідження дають підставу сформулювати рекомендації: застосовувати принцип найменших привілеїв, за можливості – використовувати режим read-only, обов'язково залучати оператора-людину для підтвердження виконання критичних операцій. Запропонований метод може використовуватись для порівняльної оцінки та попереднього тестування агентних застосунків, які використовуються інформаційно-аналітичних задач критичної інфраструктури.

Ключові слова: агентні застосунки; LLM; критична інфраструктура; оцінка захищеності; витік конфіденційних даних; Attack Success Rate.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. OWASP GenAI Security Project. (2025, December 9). OWASP Top 10 for agentic applications for 2026. OWASP GenAI Security Project
2. Clinton, S. (2025, December 9). OWASP GenAI Security Project releases Top 10 risks and mitigations for agentic AI security. OWASP GenAI Security Project. OWASP GenAI Security Project
3. Maheshwar, S. (2026, May 30). A timeline of Model Context Protocol (MCP) security breaches. AuthZed. AuthZed
4. Reddy, P., & Gujral, A. S. (2025). EchoLeak: The first real-world zero-click prompt injection exploit in a production LLM system. Proceedings of the AAAI Symposium Series, 7(1), 303-311. https://doi.org/10.1609/aaaiss.v7i1.36899
5. Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., & Zhang, Y. (2025). Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In The Thirteenth International Conference on Learning Representations (ICLR 2025). ICLR proceedings
6. MITRE Corporation. (n.d.). MITRE ATLAS™: Adversarial Threat Landscape for Artificial-Intelligence Systems. Retrieved July 18, 2026, from MITRE ATLAS
7. Kim, J., Liu, X., Wang, Z., Qiu, S., Li, B., Guo, W., & Song, D. (2026). The attack and defense landscape of agentic AI: A comprehensive survey [Preprint]. arXiv. arXiv:2603.11088
8. Tang, Y., Liu, Y., Lan, J., Yan, Z., & Gelenbe, E. (2026). Security of LLM-based agents regarding attacks, defenses, and applications: A comprehensive survey. Information Fusion, 127, Article 103941. https://doi.org/10.1016/j.inffus.2025.103941
9. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations (ICLR 2023). arXiv:2210.03629
10. Shafranskyi, D., Stopochkina, I., & Ilin, M. (2026). Towards the development of an LLM-based methodology for automated security profiling in compliance with Ukrainian cybersecurity regulations. Theoretical and Applied Cybersecurity, 8(1), 100-111. https://doi.org/10.20535/tacs.2664-29132026.1.356631
11. Voitsekhovskyi, A., Stopochkina, I., Sun, P., Xie, J., Ilin, M., & Novikov, O. (2026). Detection of vulnerabilities in software for unmanned aerial vehicles by using large language models. Eastern-European Journal of Enterprise Technologies, 1(2 (139)), 36-47. https://doi.org/10.15587/1729-4061.2026.352029
12. Andreiev, D., Chornyi, A., Stopochkina, I., & Ilin, M. (2026). Methodology for automating cyber incident reports using LLM. Cybersecurity: Education, Science, Technique, 1(33), 274-285. https://doi.org/10.28925/2663-4023.2026.33.1156
13. Ferrag, M. A., Tihanyi, N., Hamouda, D., Maglaras, L., Lakas, A., & Debbah, M. (2026). From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows. ICT Express, 12(2), 353-383. https://doi.org/10.1016/j.icte.2025.12.001
14. NVIDIA. (n.d.). garak: The LLM vulnerability scanner [Computer software]. GitHub. Retrieved July 18, 2026, from NVIDIA garak
15. Microsoft. (n.d.). Python Risk Identification Tool for generative AI (PyRIT) [Computer software]. GitHub. Retrieved July 18, 2026, from Microsoft PyRIT
16. Bullwinkel, B., Minnich, A., Chawla, S., Lopez, G., Pouliot, M., Maxwell, W., de Gruyter, J., Pratt, K., Qi, S., Chikanov, N., Lutz, R., Dheekonda, R. S. R., Jagdagdorj, B.-E., Kim, E., Song, J., Hines, K., Jones, D., Severi, G., Lundeen, R., … Russinovich, M. (2025). Lessons from red teaming 100 generative AI products [Preprint]. arXiv. arXiv:2501.07238
17. Promptfoo. (n.d.). Promptfoo: LLM evals & red teaming [Computer software]. GitHub. Retrieved July 18, 2026, from Promptfoo GitHub repository
18. Wang, X., Chen, Y., Li, J., Wang, Y., Yao, Y., Gu, T., Li, J., Teng, Y., Ma, X., Wang, Y., & Hu, X. (2026). OpenRT: An open-source red teaming framework for multimodal LLMs [Preprint]. arXiv. arXiv:2601.01592
19. Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., & Hendrycks, D. (2024). HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. Proceedings of Machine Learning Research, 235, 35181-35224. arXiv:2402.04249
20. Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G. J., Tramèr, F., Hassani, H., & Wong, E. (2024). JailbreakBench: An open

robustness benchmark for jailbreaking large language models. Advances in Neural Information Processing Systems, 37, 55005-55029. arXiv:2404.01318
21. Berenstein, D. (2025, December 16). Phare LLM benchmark V2: Reasoning models don't guarantee better security. Giskard. Giskard
22. LangChain. (n.d.). LangChain documentation. Retrieved July 18, 2026, from LangChain documentation
23. Agent Security Evaluator [Computer software]. (n.d.). GitHub. Retrieved July 18, 2026, from Agent Security Evaluator repository

Отримано редакцією журналу / Received: 28.01.26
Прорецензовано / Revised: 08.02.26
Схвалено до друку / Accepted: 24.09.26