

Cyber Response Automation with Dedicated AI Agents [†]

Stanimir Kabaivanov *  and Veneta Markovska

Faculty of Economic and Social Sciences, Plovdiv University “Paisii Hilendarski”, 24 Tsar Asen Street,
4000 Plovdiv, Bulgaria; veneta.markovska@uni-plovdiv.bg

* Correspondence: stanimir.kabaivanov@uni-plovdiv.bg; Tel.: +359-32-261-417

[†] Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

Abstract

Proactive cyber security has moved from being an innovative approach to an essential requirement for business success, especially considering the large number of new tools and technologies that organizations need to learn and use. In this paper, we discuss and demonstrate the possibility of automating important cyber security and immediate response steps with the use of dedicated artificial intelligence agents. We focus on local-first solutions that are able to keep sensitive data private and at the same time fit well with existing data protection policies and infrastructure. Using a minimalist AI agent addressing a local large language model, we experiment with skills aimed at running cyber security tools and processing their output in support of cyber incident response.

Keywords: proactive cyber security; MCP agents; automated incident response; tool orchestration with MCP

1. Introduction

New technologies cause rapid changes in the way organizations work, bringing improvements in decision making processes and increasing the amount of information to process and comprehend. While additional details enhance the transparency of typical workflows, they also bring extra complexity and increase the possibility of information system failures. These failures, whether they are purely technical in nature or a result of calculated external actions, are very costly and influence the sustainable development of organizations. There are at least two reasons for these impacts:

- Information systems are more than just a convenience, as their proper use and efficient implementation can be an important competitive advantage that complements and enhances modern management practices.

Contemporary businesses depend a lot on their data and this represents an important vulnerability, as compromising the data eventually results in a significant loss.

- Business-related data itself is crucial for success—even if it does not fall under the typical “trade secret” or “intellectual property” categories.

These reasons highlight the importance of proactive cyber security as a way to mitigate at least some of the risks associated with data breaches and information systems failures. Ref. [1] argues the importance of creating a whole culture of proactive behavior in order to be successful against ever-evolving threats, while Ref. [2] stresses the discovered gaps and deficiencies in the identification of potential threats.



Academic Editors: Nikola G. Shakev,
Valyo N. Nikolov, Nikolay
R. Kakanakov and Sevil
A. Ahmed-Shieva

Published: 3 August 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Figure 1 shows a typical proactive cyber security flow, where, due to the evolving threats and attacks, it must be executed continuously and without any interruptions, as this would otherwise make the organization vulnerable. It should be noted that even though different stages are shown as sequential steps, it is possible to have them overlap in time and be carried out simultaneously. In Ref. [3], special attention is paid to the agile nature of proactive security, which allows a reduction in damage and the elimination of certain threats before they have been able to develop into significant risks.



Figure 1. Proactive cyber security stages and typical flow.

The threat intelligence stage is responsible for analyzing the environment that organizational information systems are positioned in, in order to assess and anticipate potential vulnerabilities and the ways in which they can be exploited. While we often focus on the technical side of potential attacks, the organizational structure and preparedness of the human resources are just as important. Threat intelligence can be enhanced with various machine learning methods, as shown in Ref. [4], and artificial intelligence tools, as discussed in Ref. [5].

Continuous monitoring aims to detect issues as early as possible, which is only feasible if constant observation of the operation and data exchange is in place. As a result of any detected anomalies, further analysis needs to take place and automatic alerts should be created in the case of relevant findings. Considering the organizational and management side of the monitoring, this stage also includes building behavioral patterns for human and machine “actors” alike in order to be able to find meaningful deviations for standard operation. In this context, monitoring is not passive but has to adapt to the changing landscape of the evolving threats. Ref. [6] focuses on the application of ARIMA and time series for improving monitoring efficiency. We can take this one step further and use an ensemble of tools that are suited for various monitoring use cases. This is beneficial, as various event types have different characteristics and time series are not always the best approach for discovering them.

Vulnerability handling involves periodic scanning for possible issues and taking prompt actions to fix all detected problems. Despite this, even thorough periodic scanning and quick patching are not sufficient due to the fact that some weaknesses may not be publicly announced or may not even be found. With regard to processes and human interaction as part of organizational security, vulnerability handling involves the regular review of existing regulations and the provision of updates that are relevant to new threats.

The importance of automatic vulnerability handling is discussed in more detail in Refs. [7–9]. To address the issue that intelligent tools can be used both for protection and for the implementation of new attacks, we focus on the use of locally run systems.

The validation and testing stage includes the regular “probing” of present security measures in order to estimate their efficiency and to see if there are any deficiencies and weaknesses that need to be addressed. The non-technical part of this stage involves dedicated training and regular simulated drills for the organization’s security personnel.

The incident response stage is needed because no matter how sophisticated the implementation is made via the other proactive stages, there is no absolute guarantee that there will be no breaches or unauthorized access. In such situations, it is essential to have prepared plans that are strictly followed and regularly updated as threats and attack types evolve. Typically, this stage also requires organization staff to be extensively trained in order to make sure that existing plans are well known, understood and put into action when needed.

All of the five stages shown in Figure 1 can be automated, albeit not in full; providing an adequate level of computerization of the various steps is not sufficient due to the following reasons:

- As threats evolve, automated actions need to be reviewed and updated to make sure they are still relevant to the changing outside world and attack types.
- As organizational structures, processes and needs change, so should the automated actions—this is to ensure that they are really helping the sustainable development of the organization and not limiting it just because these processes are easier to automate.

To build better proactive cyber security automation tools, it is essential to provide a convenient and powerful way to combine human decision making and intelligence with adaptive algorithms and large knowledge bases [10]. However, building such a system requires a lot of specialized knowledge and effort to maintain solutions and keep them up to date. We explore how such resource- and time-consuming tasks can be supported by recent developments in artificial intelligence and particularly by large reasoning models.

2. Materials and Methods

Large language models are natural candidates for improving the interaction between human cyber security experts and existing dedicated tools that automate various stages of proactive security circles. Their flexibility and the enormous knowledge base that they rely upon can support security experts in analyzing traffic, studying incident characteristics and building a proper and timely response. While traditional LLMs are very good for general-purpose tasks and predicting token sequences, proactive cyber security applications require more integrity and a causal approach, which large reasoning models (LRMs) excel at [11].

Languages models are often used to support network and information security tasks, as discussed in Ref. [12]. For simple automation options, it is possible to utilize smaller models, as human operators typically break down the problem and guide the solution through individual steps in these instances. In this case, a chain-of-thought approach [13,14] fits well into the general workflow. We have opted for a model from the Qwen3 series, as this choice allows us to test its performance in both thinking (typically with a deep chain-of-thought) and non-thinking modes. This flexibility can help in separating usual tasks, which require more execution speed from complex analysis, where it is important to spend more time on causality and internal dependencies.

Unlike other LLM/RLM use cases, there are two specific requirements that are crucial for cyber security applications:

- The time needed to build a relevant response is crucial. Providing a logically correct but very late answer can be just as useless as providing a totally wrong one.

- Consistency and predictability of the response is very important for building trust in the tools deployed and avoiding accidental data loss or corruption.

To meet these requirements, we use a minimalistic system, as shown in Figure 2, that relies upon a local large language model, a custom MCP (Model Context Protocol) server and dedicated cyber security skills.

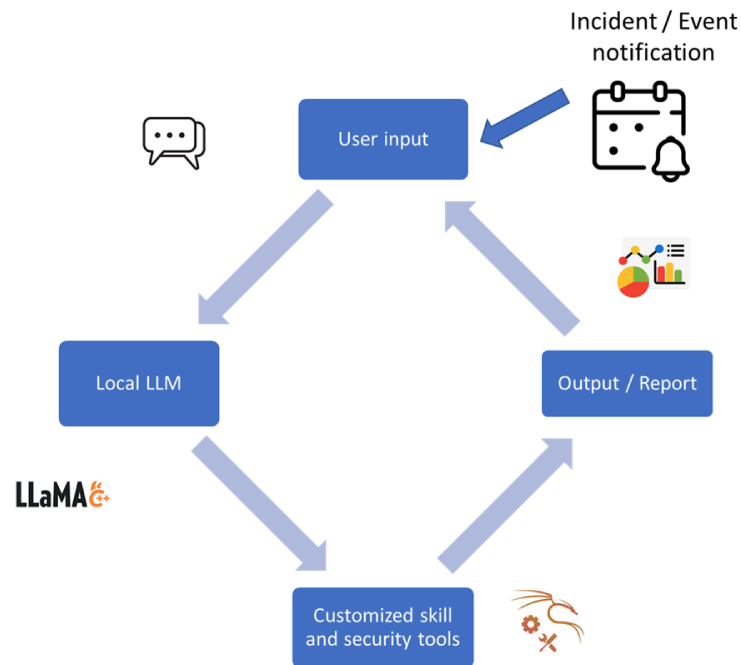


Figure 2. Minimalistic system for supporting cyber security experts.

User input in plain language can be triggered either on demand or by an incident report or detection of a custom event. This is an important feature of the whole system, as it enables use in both regular maintenance, check routines, or in immediate incident handling.

The local LLM is very important for keeping all data and communication private as it also enables organization-specific documents and rules for handling special situations to be embedded.

Customized skill and security tools are responsible for handling the cyber security tasks and represent steps like scanning, patching, performing backups or restricting access in the case of an emergency. These tools are well-known and widely used, and they are unique in the fact that invoking them and handling their input is through intelligent agent skills. Output and reporting are responsible for summarizing the output from previous step and putting it together in a useful form so that a human expert can use the results and choose what the next instructed action/sequence should be.

3. Results

To assess the impact of reasoning models on cyber security applications, we have created a simple artificial network with monitored Linux nodes. This setup contains MCP servers that comprise several standard scanning and log checking tools. While it is possible to integrate more robust IDS/IPS systems in the setup, keeping the demonstration simple allows us to focus on the integration of reasoning models with tools rather than on implementation details. We have carried out several experiments with the use of the software tools described in Table 1. As discussed previously, these tools were not selected on account of being the most powerful reasoning models or the most sophisticated security tools. Rather, they were selected due to their flexibility and simplicity of use,

making it possible to tune the final implementation depending on individual use cases and requirements.

Table 1. Tools and components used for building the experimental system.

Tools/Components	Remarks/Details
Incident reporting	For simulating an incident report or special event, we have experimented with two cases—log monitoring with tools like Graylog or logcheck and network traffic monitoring with tcpdump.
Local LLM	As a local large language model, we have used llama.cpp.
LLM Model	We have used the Qwen3.5-4B-Q4_K_M model as a compromise between size and performance. It has shown satisfactory performance, even on systems that rely solely on the CPU.
Customized skills and AI agent	We have experimented with nanobot and custom skills in order to demonstrate how security tools can be integrated.
Security tools	For the security tools demonstration, we used a subset of standard Kali Linux tools.
Output format	The output format in the experiments is a simple text output.

Figure 3 provides an overview of the architecture used for the experiments. We have built dedicated MCP servers for individual tools, separated into the following basic categories:

- Servers representing tools that are used to monitor system events, log files and network traffic.

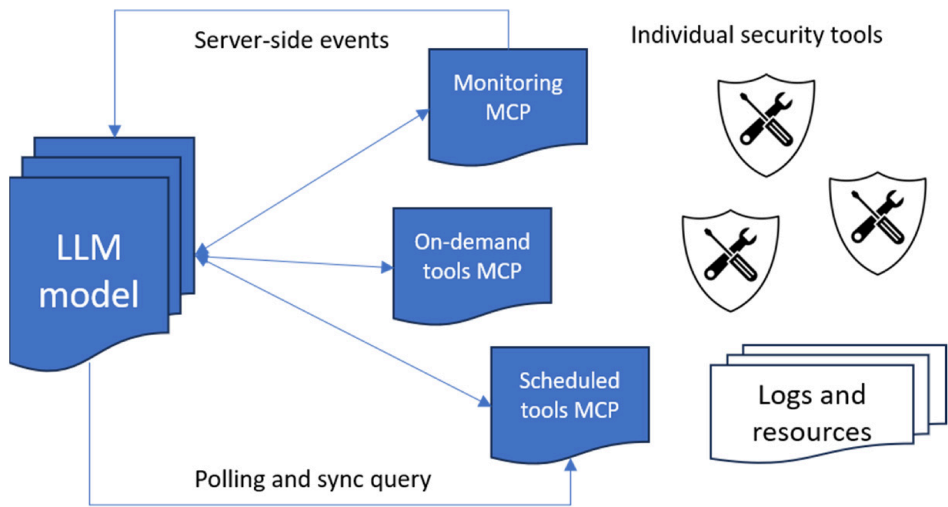


Figure 3. Experimental design and used software components.

Tools that are used to monitor system events and operation status support both synchronous query by the connected large language model and server-side event (SSE) notifications. This is needed in order to make sure that use cases can cover situations where human operators retrieve information on-demand, as well as in situations where systems bring up notifications and/or stream data continuously.

- Servers representing tools that can be invoked on-demand or based on a predefined schedule.

This group of model context protocol servers are used to execute actions based on queries and requests that are inputted by the user. They serve as wrappers to tools that are widely used to handle cyber security tasks, with the advantage of freeing human operators from having to fully understand individual options and to read out tool-specific outputs.

Representative instances from both categories can exist independently, but it is their combination that provides especially useful results. This is due to the fact that such mixtures allow the continuous monitoring of the running systems and react based on the discoveries made during these scans.

A large language model, as shown in Figure 3, is used as the front-end of the whole system and is responsible for handling natural language inputs and processing their relevant parts, resulting in the invocation of the respective MCP tools and parsing and formatting their output.

A distinctive feature of the selected architecture and set of software components is that it can be scaled to fit the complexity of the respective IT infrastructure and automation requirements. Depending on the available hardware and maximum response times, different models can be used to match the requirements. There are several important features of the software architecture presented in Figure 3:

- The role of the reasoning large language model is set to orchestrating different extensions (MCP servers that wrap up individual security tools), formatting the input and output and supporting a logical chain of steps that can automate more complex analysis and cyber threat responses.

It has to be noted that the analysis of each raw input is not the main purpose of the LLM/RLM. This decision sacrifices the flexibility and the power of the large model, as it is not used to analyze all the collected data. However, the trade-off is that the whole system gains in terms of performance, consistency and stability. The main reason behind this is that we can utilize all of the mature and highly optimized tools for cyber threat analysis, and once they have completed their work, we are able to focus on summarizing and linking their results together.

- The performance of the automated cyber security analysis and response handling is mostly influenced by the efficiency of the individual tools that are wrapped up and used.

The performance penalty that we have experienced, compared to running the tools directly, is limited by the time needed to handle natural language inputs, invoke the respective MCP server and format the outputs.

- Owing to the support for distributed MCP wrappers, it is possible to run various tools or different systems, which increases the amount of data that can be processed by the whole system.

Real-time system monitoring is a very important characteristic since it does not assume that large language models should handle all the input data—which in many cases can be measured in gigabytes. Only events of special interest or anomalies detected by less-demanding algorithms are sent for further analysis and response.

The use of various dedicated MCP servers that wrap up existing tools brings another important benefit—the ability to integrate algorithms of various complexities and speeds. This essentially turns the whole system into a hybrid solution that combines large language models with state-of-the-art machine learning and statistical analysis methods.

Details on software setup and configuration options that have been used in the experiments can be found in Table 2. As defined in the Section 1, special care has been given to the possibility of running a local LLM in order to reduce the possibility of leaking out sensitive information like system design, security keys and credentials, or network logs.

Table 2. Software setup and configuration of experiments.

Tools/Components	Configuration Details
Security Tools	Standard tools like rkhunter were used for the simple tests, where network monitoring and capture was done with tcpdump and wireshark. Each security tool runs in a dedicated environment, where sensitive information required to carry out tasks is put in restricted environment/configuration settings. This makes it possible to avoid providing security details to the LLM directly.
Local LLM	A locally running instance of llama.cpp was used, where experiments were carried out with and without GPU support (provided with Vulkan API [15]).
MCP Framework	FastMCP framework [16] was used to build Python (version 3.12) servers for individual tools and tests.
OS and Environment	Different tests were run on Linux Debian hosting the LLM and MCP.

Figure 4 demonstrates the initial results of providing a simple prompt to the reasoning language model when scanning one of the monitored systems (running a self-hosted collaboration platform) for rootkits. With reference to Figure 3, this is an example of an on-demand request, where an examination of the target system is initiated by the human operator. When triggering a complete system scanning, the invoked MCP tool does not require a lot of parameters, although the test allows various options to be set in order to control individual tests.

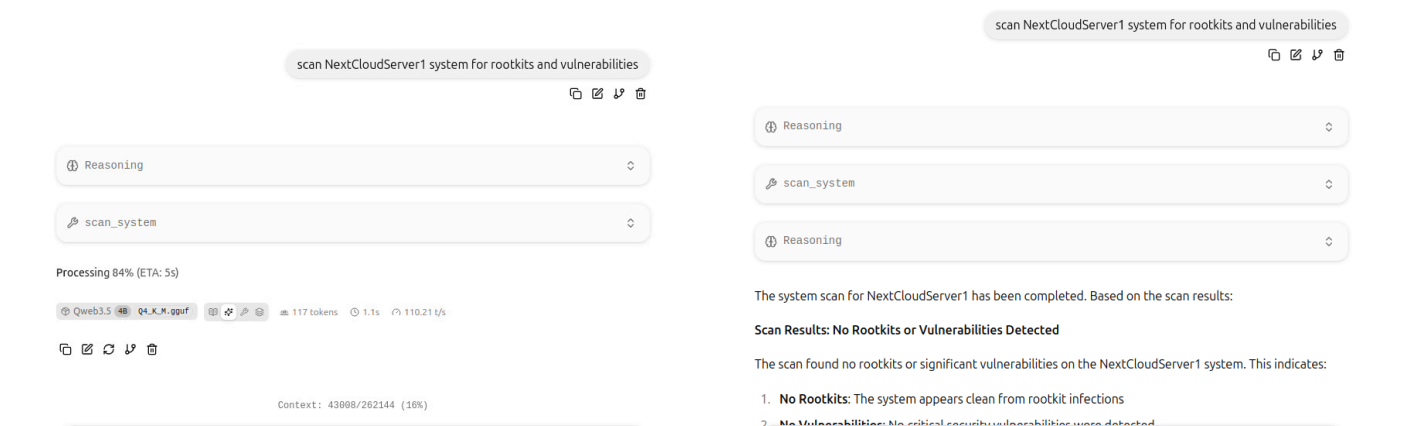


Figure 4. Sample results of the scan, including sections of scan tool output.

The right side of Figure 4 demonstrates how the reasoning language model is able to summarize the findings of the scan, which in the case of no threats is marked as clean. When there are some recommendations, the output is formatted accordingly, as shown in Figure 5.

Based on the recommendations and the required level of automation, it is possible to make changes to the running systems with the help of dedicated agents. However, this falls beyond the scope of the performed tests and requires a more careful review of suggested modifications. Considering the sensitive information that security analysis deals with, allowing fully automated changes in the configuration files and software options carries a high risk and should not be applied without very careful consideration of the consequences.

While wrapping up individual security tools is useful and offloads the burden of keeping technical details in the dedicated MCP servers, the biggest advantage of automation lies with the possibility to chain them into specialized workflows. Pipeline or wrapper servers can be used to then carry out environment-specific sequential analysis and reac-

tions that can further improve response times and reduce the need for manual steps and operation-provided inputs.

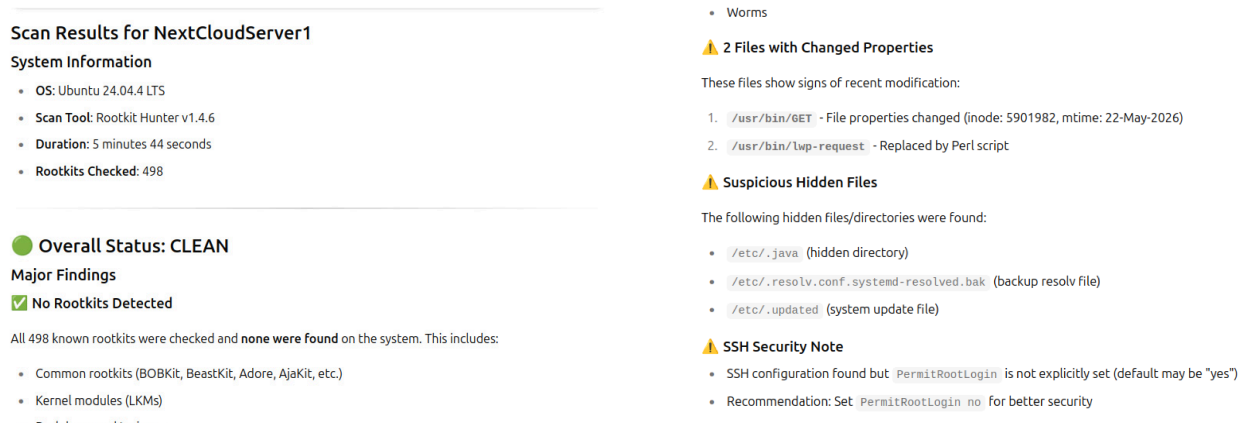


Figure 5. Summary of scan results sections and recommendations.

4. Discussion

We have demonstrated how reasoning large language models can be extended to perform cyber security-related tasks while using hardware capabilities that are commonly used and do not require significant upfront investments. The simplicity of the suggested solutions not only allows all the components to be run in a restricted/local environment, but also makes it possible to easily maintain different software components. This also follows on from the fact that reasoning models and the MCP servers attached to them can be scaled and configured in accordance with individual needs and requirements.

Using local models has both advantages and drawbacks. Being able to restrict the access to the model improves security, as it minimizes the possibility of its misuse. At the same time, MCP servers can be implemented with additional security measures, requiring authentication and restricting access to specific functionality. The fact that individual tool wrappers can be configured independently from the reasoning language model also offers security advantages, since sensitive information (like authentication details, security tokens and keys) do not have to be shared with any third party or kept within reach of the LLM/RLM.

However, running local models comes at a price—they have to be maintained and regularly improved. While such an investment of time and resources is costly, it also provides organizations with clear incentives for building efficient security procedures.

Our findings can be extended further to assess the efficiency of workflow orchestration when the reasoning model is responsible for automatically selecting the best chaining options based on the context and events found. What makes the cyber security context special is that the cost of potential errors can be very high; chained tools should intentionally keep a limited amount of sensitive information on the system in order to prevent accidental leaks.

5. Conclusions

The introduction of powerful large language models equipped with reasoning algorithms can significantly improve the way we handle cyber threats and the way organizations plan and maintain their information security systems. As attacks are getting more sophisticated, countering them requires deep knowledge and proficiency in using various tools. Keeping up with the evolving threats can be very costly and time consuming. This builds up additional pressure on smaller organizations that cannot afford to support large and specialized security teams. We have demonstrated how artificial intelligence can be used to

support and improve cyber security processes. Our study and experiments have focused on creating a hybrid system with a reasoning large language model and a set of MCP servers comprising existing security tools. Such an approach has benefits as it is easy to trace individual steps when countering cyber threats, while at the same time users have the flexibility to use natural language requests and avoid the complexities of addressing tools separately.

Considering the important and sensitive nature of the data involved in cyber threats and incidents, our suggested approach also offers two important advantages—the capability to run with locally deployed language models and full transparency, the latter of which allows users to understand and gradually improve all the included processes and steps. This adds another layer of flexibility, where specific solutions can be built that balance complexity, the degree of automation and human control.

Author Contributions: Conceptualization, S.K. and V.M.; methodology, S.K.; validation, S.K. and V.M.; writing—original draft preparation, S.K. and V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request due to restrictions, as it contains live system security scan results.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kuraku, S.; Kalla, D.; Samaah, F.; Smith, N. Cultivating proactive cybersecurity culture among IT professional to combat evolving threats. *Int. J. Electr. Electron. Comput.* **2023**, *8*, 1–7. [\[CrossRef\]](#)
2. Alqudhaibi, A.; Albarrak, M.; Alooseel, A.; Munshi, A.; Alsharif, T.; Jagtap, S.; Salonitis, K. Proactive cybersecurity in industry 4.0: A survey of cybersecurity threat prediction approaches in manufacturing systems. *Int. J. Inf. Secur.* **2025**, *25*, 14. [\[CrossRef\]](#)
3. Odeyar, S.V.; Thejaswini, K.M.; Lolakshi, P.K.; Chaithra, K.N. Reactive versus Proactive Cyber Security and Real-Time Threat Protection. In *Cyber Security in Business Analytics*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2025; pp. 61–81.
4. Bai, M.; Fang, X. Machine learning-based threat intelligence for proactive network security. *Integr. J. Sci. Technol.* **2024**, *1*, 1–11. [\[CrossRef\]](#)
5. Sarker, I.H. *AI-Driven Cybersecurity and Threat Intelligence*; Springer Nature: Cham, Switzerland, 2024.
6. Farooq, M.; Khan, M.H. AI-Driven Network Security: Innovations in Dynamic Threat Adaptation and Time Series Analysis for Proactive Cyber Defense. *Integr. J. Wirel. Microw. Technol.* **2024**, *2*, 17–26. [\[CrossRef\]](#)
7. Jena, J. Automating Vulnerability Management, the Key to Proactive Cyber Defence. *Int. J. Innov. Res. Sci. Eng. Technol.* **2025**, *14*, 1962–1969.
8. Mohammed, A. The Paradox of AI in Cybersecurity: Protector and Potential Exploiter. *Balt. J. Eng. Technol.* **2022**, *2*, 70–76.
9. Sontan, A.D.; Samuel, S.V. The intersection of Artificial Intelligence and cybersecurity: Challenges and opportunities. *World J. Adv. Res. Rev.* **2024**, *21*, 1720–1736. [\[CrossRef\]](#)
10. Kabaivanov, S.; Markovska, V. Hybrid deep-learning analysis for cyber anomaly detection. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *878*, 012029. [\[CrossRef\]](#)
11. Fu, Z.; Bao, G.; Zhang, H.; Hu, C.; Zhang, Y. Correlation or Causation: Analyzing the Causal Structures of LLM and LRM Reasoning Process. *IEEE Trans. Audio Speech Lang. Process.* **2026**. [\[CrossRef\]](#)
12. Xu, H.; Wang, S.; Li, N.; Wang, K.; Zhao, Y.; Chen, K.; Wang, H. Large language models for cyber security: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.* **2025**. [\[CrossRef\]](#)
13. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837. [\[CrossRef\]](#)
14. Zhang, H.; Lu, H.; Chen, Y.; He, Y.; Tian, Z. COAPT: Bridging Semantic-Operational Divide in Autonomous Penetration Testing Through LLM-Driven Cognitive Planning. *Big Data Min. Anal.* **2026**, *9*, 788–804. [\[CrossRef\]](#)

15. The Khronos® Group Inc. Introduction. Available online: <https://docs.vulkan.org/spec/latest/chapters/introduction.html> (accessed on 1 May 2026).
16. PrefectHQ. FastMCP. Available online: <https://gofastmcp.com/getting-started/welcome> (accessed on 2 May 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.