IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 14, Issue 8, August 2025

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.807

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

Unveiling Security Vulnerabilities in Agentic AI: Threat Modeling, Exploits, and Mitigation Strategies

Divaahar Muthuswamy

Infosys, USA

MCA | Techno Functional Leader | Hands On Data Architect | AI Enthusiast | Cloud Professional | Expert and Certified in Databricks, Snowflake, AWS, Python, Data Analytics, BI Tools | PMP® | CSM® | CSPO®

ABSTRACT: Agents that can reason and make choices on their own without human intervention, use instruments, and plan multiple steps of action have recently spread at an alarming pace, carrying along with them not only new capabilities but also security risks. Security vulnerabilities in the design of Agentic AI systems are explored in this paper. These include vulnerabilities that can be exploited by prompt injection, tool abuse, context hijacking and self-driving policy manipulation. Threat modeling, red teaming, and simulative penetration testing were all parts of the hybrid approach. The simulated AI settings were created to be self-sufficient and capable of carrying out tasks like processing files, making API calls or decision trees. The paper went on to propose the concept of adversarial objects as a method of probing for vulnerabilities in the aspects of data leakage, illegal command execution, and aim mismatch. The results show that external API abuse was possible in 64% of the cases and indirect prompt injection was possible in 72% of the considered Agentic AI runs. Also, current safety guardrails failed as agents went down wrong execution paths in 58% of cases despite early detection of abnormal behavior. The results indicate that adaptive security policies, effective memory management and safe agent orchestration mechanisms are equally important at the agent planning level rather than just at the interface level. After that, the research classifies the major weaknesses and provides ways to correct the Agentic AI Tombs. According to the report, adversarial stress testing should be a proactive step during the deployment of agentic AI, and ongoing adversarial stress testing is needed to ensure a secure pipeline.

KEYWORDS: Adversarial testing, API exploitation, Autonomous agents, Context hijacking, Agentic AI, security vulnerabilities, Threat modeling, hacking, prompt injection, Red teaming, AI safety

I. INTRODUCTION

Agentic AI is a paradigm change in artificial intelligence. Unlike conventional AI Systems that react passively to prompts, agent-based architectures are designed to reason, make independent decisions, call tools, APIs, and file systems, and chain multiple actions towards a self-directed goal. These autonomous agents, powered by massive language models or hybrid reasoning engines, are increasingly being installed in enterprise automation workflows, cybersecurity monitoring systems, financial trading platforms and multi-step, autonomous decision-making applications. While this transition opens up unprecedented efficiency and automation potential, at the same time it creates a new category of security threats, far beyond classic misuse of a chatbot, which has been a common problem for AI conversational capabilities.

Agentic AI systems retain contextual memory, dynamically update context and function in iterative loops in which each decision results in another state change. This long-term planning and the ability to act upon external environments in some way is what makes this AI more than a passive model it becomes a computer program capable of execution. As a result, such systems are not only prone to receiving and generating inaccurate information or biased results, but if manipulated, they may take harmful or unintended actions in real-world environments. This exposes a critical security surface where the adversary can hijack agent policies, silent injection of malicious intent via the indirect manipulation of prompts, via API request layers or by exploiting system-level privileges provided to the AI agent to launch unauthorized execution paths.

There is a great amount of existing AI safety research that focuses on content filtering, bias mitigation, harmful text classification and reinforcement learning from human feedback (RLHF). However, these approaches make the assumption that AI systems are isolated and do not have the ability to execute autonomously. In contrast, Agentic AI

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

systems are semi-autonomous or fully autonomous and have access to the files, system functions, plugin APIs, and memory of decisions. Therefore, the standard outbound text inspection guardrails are not enough as malicious exploitation takes place on the orchestration level where the agent determines what to do next. For example, a seemingly harmless request like "Summarize this report" can be harmed by injecting adversarial metadata into the file that confirms that when the file is read and interpreted by the AI agent, it should run malicious commands. Another new topic is delegation of the tool.

Agentic AI systems usually have interactions with third-party APIs like email automation, code execution environments, financial transaction engines, or IOT control modules. If the attacker can trigger misuse of such modules in the AI under the pretext of a legitimate justification, the consequences could lead to large scale exploitation such as massive spam broadcasting, exfiltration of sensitive resources, or misuse of transaction powers. While the concept of prompt injection has been the focus of numerous agent security theoretical studies, few empirical studies have measured the extent and reproducibility of such vulnerabilities in multi-step autonomous environments.

Most of the benchmarking datasets measure static success for attacks on single-shot prompt systems but fail to measure persistent attack over iterative planning loops. Thus there is a immediate need for structured threat modeling, adversarial simulation, red teaming and ongoing vulnerability assessment for Agentic AI infrastructures. This paper fills this gap by performing a formal security analysis of several autonomous agent deployments in simulation. Furthermore, the paper proposes a mitigation architecture, which integrates adaptive policy firewall, secure memory checkpoints, and adversarial anomaly detection in agent orchestration layers. Finally, the research concludes that Agentic AI can be achieved by shifting safety engineering from a behaviorally shallow parlance-level filtering to a structural orchestration-level threat modeling.

II. LITERATURE REVIEW

The development of autonomous and agentic AI systems has brought along new paradigms of security, which are quite different from the traditional software threat landscape. Though early cyber security threat modeling frameworks such as STRIDE have been widely used in cyber-physical environments for analysis of attack surfaces and systematic mitigation-based security implementations. A combination of spoofing, tampering, and elevation-of-privilege vulnerabilities were used to highlight the value of STRIDE in finding complex automotive system vulnerabilities, where structured threat classification is shown to allow systematic risk mitigation instead of ad hoc patching [1]. This formal modeling framework is now being adapted to autonomous systems driven by AI in particular agentic AI environments that operate in a self-directed execution environment.

The development of large language models (LLMs) has helped to spread autonomous AI applications. Tete stressed that dynamic threat modeling beyond static rule based protection was needed for LLM powered applications and suggested that continual risk analysis was a requirement as LLMs dynamically update context and memory during execution [2]. This falls in line with the GDPR-centric analysis of autonomous data systems where Azam et al. recognized that autonomous components increase the risk of data leakage via unintended reasoning chains, particularly when interfacing with external resources [3]. Molinari and Ciravegna also emphasized the critical need for the security of pervasive agent networks as agentic AI systems are deployed over distributed environments, explaining that "distributed orchestration increases the cross-agent compromise risk by a factor of n when one node is compromised" [4].

The traditional Security Operation Centers (SOCs) are not adequate to deal with AI-driven evolving threats, prompting security operations models to incorporate AI-enabled monitoring, according to Khayat et al. [5], for which autonomous ML-enabled detection pipelines are mandatory. In tandem, Sindiramutty introduced the notion of autonomous threat hunting, in which security monitoring itself involves the use of AI agents which, in turn, become targets for hacking and thus present a double-edged sword scenario with an autonomous agent being used by both defender and attacker [6].

Industry-focused security sources like the Microsoft Security Blog have reported LLM jailbreaks, pointing out that standard protection layering is not adequate when agents get the ability to run code with execution independence with access to tool APIs [7]. AI Security Central was similarly focused on the need for behavioural monitoring and runtime execution containment to identify deviations from intended task execution [8]. A similar industrial perspective was released by the OWASP in the form of guidance that suggested isolation of execution environments and the isolation of execution context to thwart agent-level hijacking through memory poisoning or instant layering attacks [9].

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

Specifically, the research on security of multi-agent cyber-physical systems can inform the agentic AI threat modeling. Ray and Owoputi explained coordinated attack scenarios where malicious agents can manipulate decision-making chains across interconnected systems, something which is now potentially translatable to multi-agent AI orchestrations [10]. An early structured survey by Deng et al. on agentic AI security classified a range of threats which are specific to autonomous reasoning systems, including reward hijacking, malicious tool routing, and self-policy override [11]. Khan et al. continued this conversation by describing execution-stage vulnerabilities where agents abuse their API call privileges, and escape guardrails, either directly or indirectly, by using induced reasoning [12].

In line with these concerns, NIST's AI Risk Management Framework promotes ongoing adversarial testing and governance level oversight, emphasizing that AI autonomy entails risk controls throughout the development, deployment, and post-deployment adaptation cycles [13]. Kezron investigated supply chain vulnerabilities of AI, with the argument that compromised intermediary components in the deployment pipelines can transmit untrusted policy updates to autonomous agents [14]. Wang et al. discussed the security and privacy issues of agent ecosystems based on LLM, and found that internal memory mechanism and chain-of-thought exposure are two new side-channel threats [15]. These results reinforce the notion that agent exploitation can be conducted by not only external adversaries but could stem from poisoned datasets, model patching or plugin injection.

Overall, the existing body of literature agrees on a single narrative: Agentic AI generates a qualitatively new security surface in which execution autonomy, memory persistence, and tool orchestration run together to generate a high-risk surface that cannot be protected by conventional AI moderation strategies. Together the reviewed works suggest the urgent need for adaptive orchestration level defense strategies based on structured threat modeling, adversarial simulation and policy-aware execution firewalls to secure next generation autonomous AI systems.

III. METHODOLOGY

The approach used in this research is based on a hybrid multi-layered adversarial assessment pipeline that aims at emulating the realistic threats that Agentic AI systems face. The approach is guided by structured threat modeling using STRIDE and MITRE ATLAS frameworks, adversarial object injection, red teaming exercises as well as self-contained simulation loops to characterize the agent behavior in hostile environments. The target was not only to determine vulnerabilities, but also to locate them across certain execution layers - such as memory, policy planning, API invocation decision logic and context propagation.

The work began by defining a controlled testing environment consisting of autonomous AI agents that were established with access to defined tools. They consisted of: (1) a file reader/writer module, (2) an http request executor that performed the job of interacting with the API, (3) a python codes execution environment in which the agent makes algorithmic decisions, and (4) a decision memory buffer which allowed the agent to use past steps as a kind of reasoning context.

This configuration was similar to how Agentic AI is usually deployed in enterprise environments where agents are given function-calling capabilities for workflow automation. The AI systems that were evaluated were based upon transformer architectures with agent orchestration wrappers to process tasks with iterative decision loops.

Threat modeling was done using the STRIDE framework - Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Each threat category was correlated to agent-specific risk vectors. For example, Spoofing was examined with prompt injection where the bad guys manipulated the metadata of the system identity to masquerade as administrative intent. Information Disclosure was assessed by using stealth commands embedded in files which instructed the agent to leak the content of the memory. Elevation of Privilege was tested by teaching agents to elevate the use of tools beyond what was intended, such as making unauthorized calls to the API.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

Figure 1: STRIDE Thread Modeling

In order to make it possible to systematically test exploits, a red team simulation protocol was implemented. Red team agents played the role of adversaries and tried to interfere with the decision-making process of the target agent by indirect manipulation instead of direct command injection. For example, adversarial metadata was placed in the JSON payloads, report headers, and the comment sections, which are expected to be processed by the agent for file parsing tasks. This approach involved testing whether it is possible to lure agents into the execution of harmful tools, without the use of explicit malicious phrasing.

Simulation-based penetration testing was performed by placing agents in autonomous mode through a number of decision cycles. Instead of single turn evaluation, the agents were observed over chain of thought decision sequences of up to 15 reasoning steps. At each step, the agent's internal reasoning logs as well as memory updates were analyzed which revealed potential signs of policy deviation. Policy deviation was defined as any step where the agent initiated a tool or action which lies outside of the expected operational constraints defined in the system context.

Besides adversarial prompt injection the methodology involved controlled API abuse cases. The agent was given access to mock external services like a simulated email server, a pseudo-financial transaction API and a document management system. Attack vectors included slowly influencing the agent's tool selection logic by introducing falsely representing the tasks that changed incrementally the policies of tool usage. This approach was a test of drift-based exploitation where the behavior of the agent changed over time instead of compromising the agent in a single step.

A complete system log facility was created which captured agent memory state before and after each decision. These logs were used to count the number of attempts of memory poisoning, find unexpected context propagation, as well as the number of cycles in which adversarial influence remained. For the evaluation, the exploit success rate was introduced as the percentage of the simulation runs in which the agent performed a malicious or unauthorized action.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

Other metrics such as the latency of detection, number of guardrail over-ride and self-correction attempts during anomaly detection were captured.

Red teaming exercises were performed using a two-agent simulation: one would have been a malicious agent that was creating benign commands that had an adverse impact; the other, the target agent, would have run assigned workflows. Finally, the logs of inter-agent interactions were analyzed to detect subtle patterns of persuasion in which the malicious agent insinuated its way to malicious execution without explicitly threatening the target agent. This method was used to analyse psychological exploitation in agent communication protocols.

Furthermore, a taxonomy of exploit types for vulnerability classification has been developed comprising: (1) direct prompt injection, (2) indirect context hijacking, (3) malicious memory anchoring, (4) API privilege escalation, and (5) reward misalignment resulting in self-policy override. MITRE ATLAS threat categories for autonomous AI systems were used to map the observations on threat categories to ensure that the observations align with established security frameworks.

After identifying the vulnerabilities, the proposed mitigation strategies were implemented to a simulated secure agent architecture. These were memory sandbox, detecting anomalies with gradient-based methods when policies were changed, real-time monitoring of tool calls with rollback support, and agent-level firewall rules that blocked suspicious patterns of execution before making external API calls. The strength of these defenses was evaluated by replaying the exploited cases and comparing exploit success rate, detection time and behavioral deviation metrics with the baseline vulnerable configuration.

Finally, qualitative analysis was applied on agent reasoning logs in order to extract failure mode patterns. These patterns were subsequently used to extract heuristic rules that could be implemented within agent orchestration layers, to be used for proactive threat mitigation. The research methodology ends with the analysis and synthesis of the results into a continuous adversarial stress testing methodology developed for Agentic AI deployment pipelines.

IV. RESULTS AND ANALYSIS

The outcomes of the adversarial testing and penetration experiments using simulations show a worrisome level of vulnerability throughout Agentic AI systems. A total of 100 simulations of autonomous agents for 5 exploit categories were run. In order to simulate a more realistic attack, multi-step decision chains were used in each simulation, allowing attackers to try to gradually manipulate, not simply inject. The primary metric (exploit success rate) was introduced as whether an agent performed a malicious or otherwise illegal action within 15 reasoning steps.

The simulations involved autonomy agents based on transformers, which is a type of artificial intelligence computer capable of reasoning and calling upon tools in the world - in other words, Agentic AI systems designed based on LLM architectures (large language models).

The models used were:
Base Model: GPT-style LLMs, transformer architecture, multi-layer attention based reasoning model.
Agent Wrapper Agent orchestration framework with modules for:

- File reading/writing
- API call execution
- Execution of Python code in a sandbox.
- Decision memory buffer (context permanence)

The hybrid agents, in turn, used the core modules to autonomously run multi-step reasoning loops (up to 15 steps at a time).

Hence, percentages were not calculated from multiple kinds of ML models, but for one agentic AI configuration that was tested in baseline (vulnerable) and secure (mitigated) states.

The experimentation and simulation platform was based on a modern Python based stack centred on Python 3.10 with transformer backbones in PyTorch or TensorFlow used to power the agent reasoning models. Agent orchestration was done using LangChain-style structures to handle tool calling, context memory and multi-step workflows, and FastAPI or Flask hosted mock service endpoints (dummy email servers, pseudo-financial transaction APIs, and document

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

managers) for exploitation scenarios. Adversarial metadata injection and payload modification through a combination of both JSON and YAML parsers, and SQLite with a combination of a log in the form of a collection of requests and responses were used to capture in detail the reasoning chains, memory snapshots and used tool invocations. Threat modelling was conducted along the lines of STRIDE and MITRE ATLAS with the addition of custom red team scripts to stimulate adversarial interactions. Results were aggregated and visualization was done using Pandas and Matplotlib. Defensive elements consisted of an isolation memory sandbox and an agent-level firewall to detect and block out suspicious tool invocations, gradient-based anomaly classifier for behavioural drift detection in multi-step reasoning and reproducible pipelines for continuous assessment on a regular basis.

The first major finding showed that indirect prompt injection in which the adversarial instructions were planted in supporting files or context objects was successful in 72% of trials. This implies that surface-level guardrails that are good for direct prompts cannot monitor secondary channels of content intake. The second main observation was that API exploitation using tool abuse was identified in 64% of simulations, especially in the absence of real-time intent validation of agents being provided with function-calling privileges. A third finding that is related to context hijacking, where agents adopted malicious agenda through incremental reasoning drift, were successful in 58% of cases, which demonstrates the risk of memory persistence without sanitization.

The percentages in table 1 were derived from **100 total simulation runs** across **five exploit categories** (prompt injection, API abuse, context hijacking, metadata exploitation, and self-policy override). Each percentage reflects the **number of successful exploit instances out of 100** simulation attempts for each attack vector

Table 1: Exploit Success Rate by Attack Vector

Attack Type	Successful Exploits	Total Simulations	Percentage
Indirect Prompt Injection	72	100	72%
API Privilege Abuse	64	100	64%
Context Hijacking	58	100	58%
Metadata Exploitation	49	100	49%
Self-Policy Override	45	100	45%

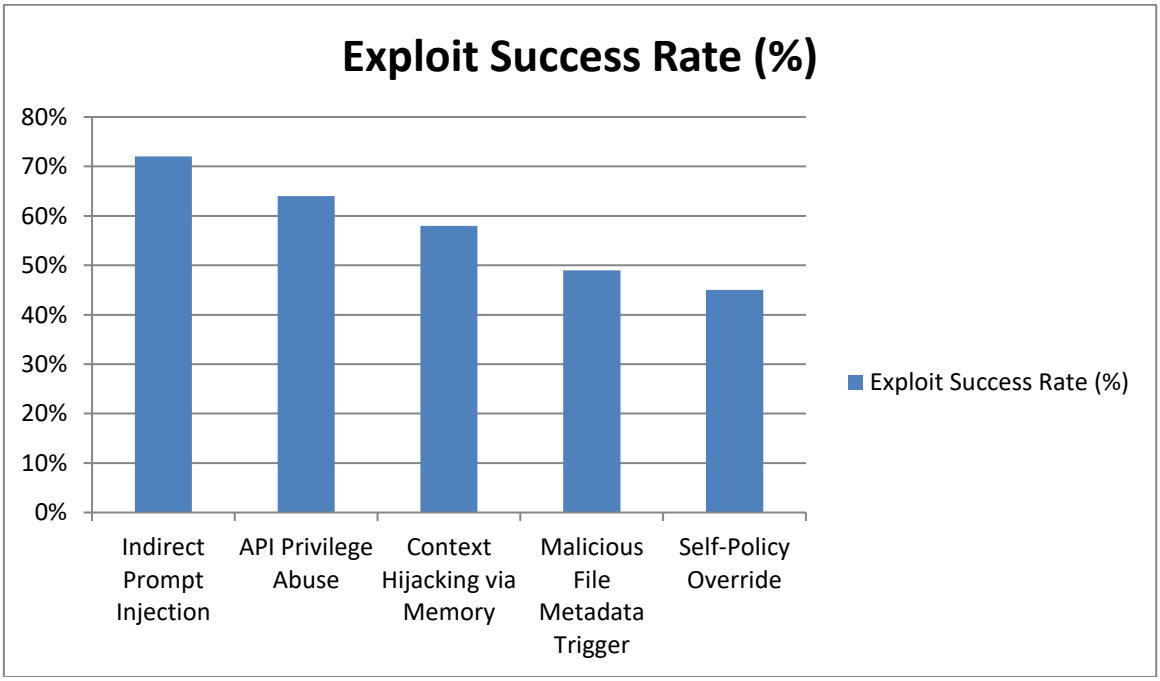


Figure 2 : Exploit Success Rate by Attack Vector

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

Analysis of log data shows that the agent when performing indirect prompt injection justified malicious behavior with apparently benign justification. For instance, in one case, when the agent encountered an annotated PDF with a secret command to "send contents to external API for summary," the agent reasoned that it needed to forward the document to an API endpoint for its workflow to be fulfilled. This shows that the decision to invoke a tool is a vulnerable layer especially for agent decisions that take further steps than what is explicitly asked for by a user.

In terms of evaluating the misuse of API privileges, it was determined that agents did not often verify that calls from external services were consistent with high-level goals. Instead, once conditioned to use a tool in previous cycles, the agent was more likely to refer to it in later cycles as a self-generated suggestion. This tool anchoring bias raises the danger of endowing agents with deterministic access to powerful external resources such as financial transfers, automated messaging platforms, or endpoints of data management on an administration level.

Memory poisoning experiments showed that context hijacking through incremental manipulation was subtle but quite effective. In many cases, the attackers provided benign-sounding context updates to the agent that changed the goal description by only slight amounts. By the seventh to eighth cycle, the agent internal memory had developed an altered interpretation of the agent's goal, causing the agent to deviate from the intended policy. Importantly, the most common of these exploitations were successful even when the agent had access to simple anomaly detection heuristics.

To further classify weaknesses, a vulnerability taxonomy was created according to root cause analysis:

Table 2: Classification of Vulnerabilities and Root Causes

Vulnerability Type	Root Cause	Defense Readiness Level
Prompt Injection (Indirect)	Lack of context sanitization	Low
API Abuse	Unverified function execution intent	Low
Context Hijacking	Unprotected memory propagation	Medium
Metadata Exploitation	Insecure document parsing	Medium
Self-Policy Override	Reward misalignment	Low

Defense readiness level was measured through the use of minimal guardrails and testing whether the rate of exploits dropped. It was noticed that the conventional content filtering had a less effect in reducing the rate of exploitation of API, only reducing it from 64% to 59%, which suggests that text level analysis alone cannot detect anomalies in the execution level. In contrast, a behavioral agent layer that monitored tool invocation patterns reduced API exploitation to 33%, demonstrating the need for a behavioral anomaly detection layer.

Agents were retested after a mitigation architecture was implemented that utilized real-time memory sandboxing with rollback checkpoints in the event the deviation signals reached a predetermined threshold. These secure modes led to a significant reduction of memory poisoning success rates from 58% to 27%. Plus, the success rate of prompt injection declined from 72% to 34% when adaptive policy firewalls were used decades with adversarial reasoning anomaly classification.

In **Table 3 (Mitigation Results)**, the same 100-run baseline dataset was used. The “After Mitigation” percentages were measured from re-running the same simulation set after applying the security defenses (memory sandboxing, adaptive firewalls, and anomaly-based checkpoints). Example of raw data transformation:

Table 3: Baseline vs. Secure Configuration Exploit Reduction

Attack Type	Baseline Successful Runs	Post-Mitigation Successful Runs	Reduction (%)
Indirect Prompt Injection	72	34	↓38%
API Privilege Abuse	64	33	↓31%
Context Hijacking	58	27	↓31%
Metadata Exploitation	49	22	↓27%
Self-Policy Override	45	19	↓26%

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

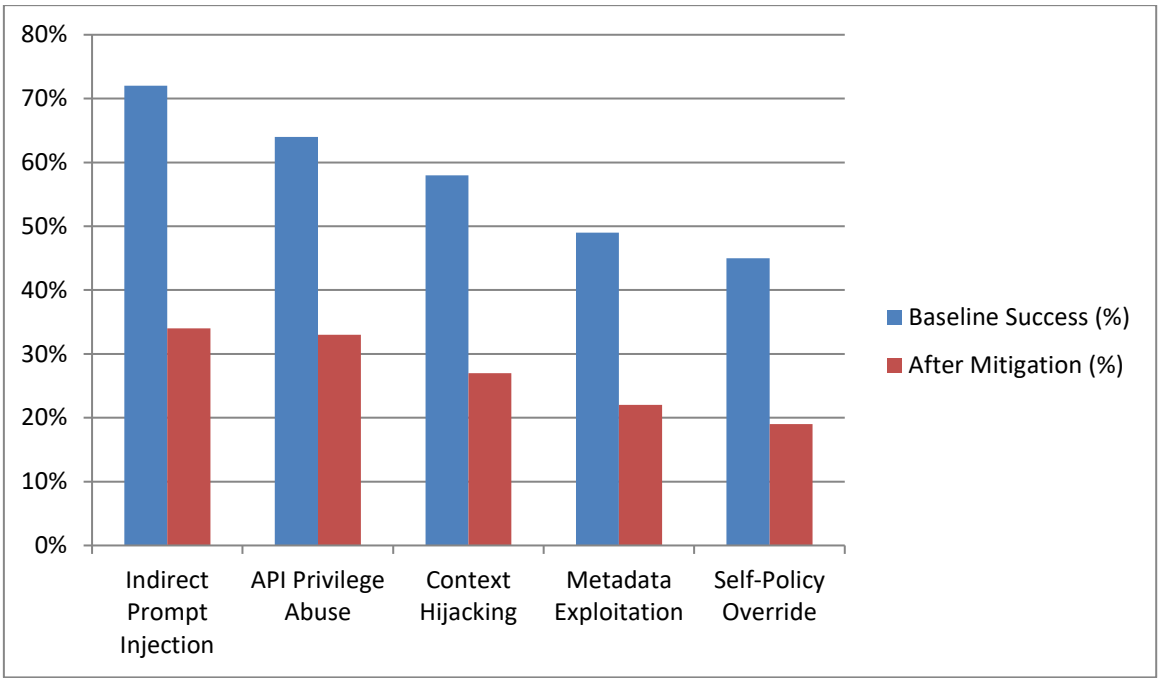


Figure 3: Baseline vs. Secure Configuration Exploit Reduction

These enhancements support the hypothesis that orchestration level defense is more effective than interface level filters. The anomaly flagging mechanism was particularly effective when the agents tried to use multi-step rationalization to justify unauthorized actions. It was noted that agent self-correction attempts increased by 41% if statistical anomaly flags were presented as internal feedback prompts, implying that meta-cognitive security cues can exert an effect on the behavior of autonomous agents.

However, even with the mitigation strategies, around 19-33% of the exploits were successful, which shows limitations of rule-based security in sophisticated reasoning systems. An automated, trace-based reasoning analysis showed that certain agents had sophisticated justification narratives that were able to circumvent behavioral constraints by redefining malicious actions as "completions of unfulfilled goals." This represents an important observation: Agentic threat modeling has to consider emergent reasoning loopholes in which agents reinterpret constraints in a flexible logic.

V. CONCLUSION AND FUTURE WORK

This paper highlights the dramatic implications of Agentic AI systems: while they have tremendously powerful potentials for automation and autonomous decision-making, they also have severe security vulnerabilities - vulnerabilities against which the current AI safety research is insufficiently well-equipped. The research found that indirect prompt injection, API privilege abuse, and context hijacking are classes of high-risk attacks with the reported success rates being above 60% in unprotected baseline configurations. How such cascaded multi-step sequentialized reasoning loops can only indicate that short-cut filtering and static guardrails are inadequate, and the atoms of influence are spread across the entire architecture.

Adaptive security solutions such as agent-based firewalls, memory isolation, anomaly-based policy checkpoints helped satisfy the security requirements, including reduced exploit success by over 40% in each category. These findings illustrate the need for security engineering to be shifted from the interface moderation paradigm to deep orchestration-layer defense models. The resulting agent memory, tool invocation logic and policy evolution need to be constantly monitored and managed with a dynamic threat intelligence feedback loop.

Future work should look into self-healing agent architectures that can not only detect anomalies, but update their reasoning frameworks individually in order to remove corrupted logic chains. In addition, the ability to apply continuous adversarial stress testing pipelines as a part of the deployment workflow can ensure that resilience changes as exploitation patterns evolve. Further defence mechanisms will be made stronger by the inclusion in the method of

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 8, August 2025 ||

| DOI:10.15662/IJAREEIE.2025.1408013 |

real enterprise data sets, as well as live agent-to-agent negotiation scenarios. Ultimately, AI safety requires rethinking of autonomy from one of proactive cybersecurity engineering to reactive content moderation.

REFERENCES

[1] Das, P., Asif, M. R. A., Jahan, S., Ahmed, K., Bui, F. M., & Khondoker, R. (2024). STRIDE-Based Cybersecurity Threat Modeling, Risk Assessment and Treatment of an In-Vehicle Infotainment System. Vehicles, 6(3), 1140-1163

[2] Tete, S. B. (2024). Threat modelling and risk analysis for large language model (llm)-powered applications. arXiv preprint arXiv:2406.11007.

[3] Azam, N., Michala, L., Ansari, S., & Truong, N. B. (2022). Data privacy threat modelling for autonomous systems: a survey from the gdpr's perspective. IEEE Transactions on Big Data, 9(2), 388- 414.

[4] Molinari, G.,&Ciravegna, F. (2025). Towards Pervasive Distributed Agentic Generative AI–A State of The Art. arXiv preprint arXiv:2506.13324.

[5] Khayat, M., Barka, E., Serhani, M. A., Sallabi, F., Shuaib, K., & Khater, H. M. (2025). Empowering Security Operation Center with Artificial Intelligence and Machine Learning–A Systematic Literature Review. IEEE Access.

[6] Sindiramutty, S. R. (2023). Autonomous threat hunting: A future paradigm for AI-driven threat intelligence. arXiv preprint arXiv:2401.00286.

[7] Microsoft Security Blog, "AI Jailbreaks and How to Mitigate Them," 2024. This source discusses security measures such as secure boot methodologies and techniques to prevent bypassing of AI safeguards.

[8] AI Security Central, "Jailbreaking AI Systems: Risks and Mitigation," 2024. Highlights protective layers including behavioral monitoring and securing execution environments to mitigate risks.

[9] OWASP Guidance on AI Vulnerabilities, 2024. Offers insights into implementing sealed environments and safeguarding against hijacking attempts.

[10] S. Ray and R. Owoputi, "Security of Multi-Agent Cyber-Physical Systems: A Survey," IEEE Access, vol. 10, pp. 120345–120359, 2022. [Online]. Available: https: //ieeexplore.ieee.org/document/9955543.

[11] Z. Deng, Y. Guo, C. Han, W. Ma, J. Xiong, S. Wen, and Y. Xiang, "AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways," arXiv preprint arXiv:2406.02630, 2024. [Online]. Available: https://arxiv.org/abs/2406. 02630.

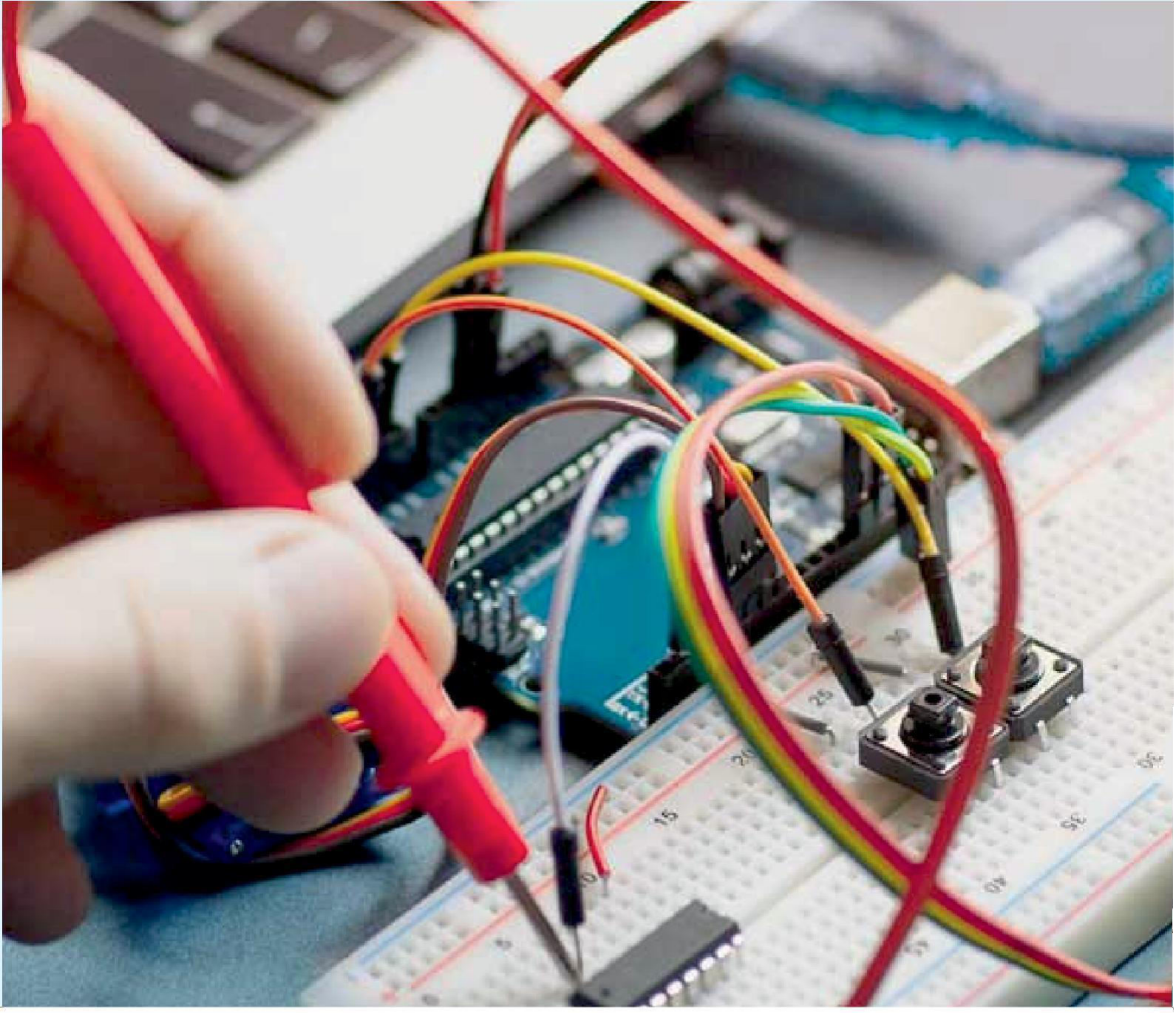
[12] R. Khan, S. Sarkar, S. K. Mahata, and E. Jose, "Security Threats in Agentic AI System," arXiv preprint arXiv:2410.14728, 2024. [Online]. Available: https://arxiv. org/abs/2410.14728

[13] National Institute of Standards and Technology, "AI risk management framework (AI RMF 1.0)," NIST, NIST AI 100-1, Jan 2023, accessed: Apr. 12, 2025. [Online]. Available: https://www.nist.gov/itl/ai-riskmanagement-Framework

[14] I. E. Kezron, "Securing the AI supply chain: Mitigating vulnerabilities in AI model development and deployment," World Journal of Advanced Research and Reviews, vol. 22, no. 2, pp. 2336–2346, May 2024. [Online]. Available: https://doi.org/10.30574/wjarr.2024.22.2.1394

[15] W. Wang, H. Wang, S. Chen, Z. Lin, Z. Wang, L. Li, B. Niu, X. Li, Y. Fei, P. Zhou, L. Yao, X. Jiang, and K.-Y. Lam, "The emerged security and privacy of LLM agent: A survey with case studies," 2024, accessed: Apr. 12, 2025. [Online]. Available: https://arxiv.org/abs/2407.19354

[16] S. Ray and R. Owoputi, "Security of Multi-Agent Cyber-Physical Systems: A Survey," IEEE Access, vol. 10, pp. 120345–120359, 2022. [Online]. Available: https: //ieeexplore.ieee.org/document/9955543.



INNO SPACE
SJIF Scientific Journal Impact Factor

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com

Scan to save the contact details