PoisoningGuard: An Agentic LLM Framework for Explainable Red-Teaming Against Poisoning Attacks[*]

Junseok Shin, Jinhyuk Jang, and Daeseon Choi [†]

Soongsil University, Seoul, South Korea
{james9907,slsk100}@soongsil.ac.kr, sunchoi@ssu.ac.kr

Abstract

As artificial intelligence systems become increasingly deployed across critical social and industrial domains, ensuring model security and reliability has emerged as a fundamental challenge. Among various threats, adversarial techniques such as data poisoning pose severe risks by corrupting the training process itself. While existing red-teaming frameworks enable automated attack simulations, they often lack sufficient explainability and in-depth analytical capabilities during the detection process. To address these limitations, this study proposes an agent-based LLM-driven security evaluation framework. The proposed system leverages the reasoning and tool-usage capabilities of large language models to integrate end-to-end processes, including program execution and log analysis, literature-informed attack detection, evidence generation, multi-turn memory management, and response quality assessment. This holistic design enables not only the detection of adversarial behavior but also ensures transparency in the reasoning process and facilitates knowledge-grounded, fine-grained analysis. Our findings demonstrate that, unlike conventional tools, the proposed framework enables automated security experiments with enhanced explainability and adaptability. Furthermore, it highlights the potential to improve the trustworthiness of adversarial attack detection and to extend toward diverse security scenarios.

Keywords: Poisoning Attack, Agentic AI, AI Security, Red-Teaming

1 Introduction

As artificial intelligence (AI) systems are increasingly deployed across critical domains such as healthcare, finance, autonomous driving, and defense, the importance of AI security has become more pressing than ever. Nevertheless, such systems remain highly vulnerable to adversarial attacks, particularly poisoning attacks that compromise the training phase by manipulating data, thereby fundamentally undermining model performance. Poisoning attacks can be realized by modifying portions of the training data or inserting adversarially crafted samples, causing the model to misclassify specific inputs. This not only degrades overall model accuracy but may also induce catastrophic failures in specific tasks or scenarios. Prior studies have proposed various frameworks for detecting and defending against such attacks. Microsoft's Counterfit [16] enables automated execution of diverse attack algorithms to identify model vulnerabilities, while IBM's Adversarial Robustness Toolbox (ART) [21] provides an integrated environment for experimenting with both attack and defense strategies. However, these tools often fall short by reporting detection results only in numerical form, lacking explainability regarding the causes of attacks and omitting multi-turn, in-depth analysis. Meanwhile, the recent advances in agentic

[*]Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 6, December 16-18, 2025, Sapporo , Japan. © The copyright of this paper remains with the author(s).
[†]Corresponding author

AI powered by large language models (LLMs) open new opportunities to address these limitations. LLMs, when combined with external tools, can perform planning, reasoning, retrieval, and code execution in a unified manner, while multi-turn interactions allow them to maintain context and deliver progressive, explainable analysis. In particular, retrieval-augmented generation (RAG) mitigates the hallucination problem by incorporating external knowledge in real time, thereby enabling reliable, evidence-grounded responses. Building on these developments, this study introduces PoisoningGuard: An Agentic LLM Framework for Explainable Red-Teaming Against Poisoning Attacks. The proposed system implements an integrated five-stage pipeline: (1) automated program execution and log collection, (2) RAG-based detection and classification of poisoning attacks, (3) literature retrieval and cross-verification, (4) multi-turn memory–driven in-depth analysis, and (5) response quality evaluation using RAGas. This design enables the framework not only to determine whether an attack has occurred but also to explain which attack type was detected, why it was classified as such, and which original research work first introduced it. Experimental evaluations under a Brainwash attack [1] scenario demonstrate that the proposed framework provides explainability and multi-turn reasoning capabilities not offered by Counterfit or ART. Moreover, quantitative assessment with RAGas metrics confirmed the reliability and relevance of the results, yielding a faithfulness score of 0.73 and an answer relevancy score of 0.77. These findings indicate that LLM-based agents can play a pivotal role in AI security assessment and red-teaming activities. The contributions of this paper are summarized as follows:

- We propose a novel framework that automates the end-to-end process of poisoning attack detection through agentic AI powered by LLMs.
- Unlike existing frameworks such as Counterfit and ART, our approach provides explainability of attack causes and supports multi-turn, in-depth analysis.
- We empirically validate that combining RAG with RAGas enables quantitative evaluation of reliability, explainability, and response quality in attack detection results.

2 Related Work

2.1 AI Agent and Tool Use

AI agents are intelligent systems capable of autonomously perceiving feedback, engaging in iterative interactions, and performing real-time decision-making within dynamic environments. Merely generating outputs from textual inputs through an LLM is insufficient to qualify as an AI agent; rather, an agent requires an architecture in which the LLM functions as the “cognitive core,” enabling multimodal information processing alongside planning and reasoning [9]. For such agents to achieve optimal performance, they must be able to strategically select and exploit the most appropriate tools from a diverse set of external resources.

Bhargavi et al [**23**] proposed Automatic Reasoning and Tool use, a framework in which an LLM decomposes a new task into multiple steps and, at each step, strategically selects and applies external tools such as calculators, search engines, or code executors. Automatic Reasoning and Tool use outperforms conventional Chain-of-Thought (CoT) prompting [26] in complex problem-solving by retrieving similar cases from a task library, controlling intermediate reasoning through a structured query language, and incorporating tool outputs into subsequent inference. This approach achieved an average performance improvement of 22 percentage points across diverse benchmarks, including BigBench and MMLU, and surpassed the prior GPT-3 state of the art by 6.1 percentage points on arithmetic and algorithmic tasks. Moreover, task- and tool-library updates allow rapid incorporation of human feedback, enabling efficient enhancement of task-specific performance.

Jakub et al [17] introduced PaperQA, an agent-based RAG system designed to overcome the limitations of conventional LLM-based question answering, such as hallucination, lack of access to up-to-date information, and delayed verification. PaperQA modularizes the workflow into three stages—paper retrieval, content extraction, and answer generation with references—while allowing the agentic LLM to dynamically adjust retrieval and reasoning strategies according to query characteristics. This framework outperformed GPT-4 and commercial QA systems on PubMedQA and LitQA, particularly demonstrating advantages in leveraging recent knowledge and mitigating hallucination.

Na et al [19] proposed the RAISE architecture, which extends the ReAct framework by introducing a dual-memory design that combines a scratchpad for short-term memory with a retrieval module for long-term memory. This design enables agents to maintain contextual awareness and adaptability in complex, multi-turn conversations. RAISE improves reasoning accuracy by standardizing the CoT–tool use procedure and reduces unnecessary planning and action steps, thereby enhancing response efficiency. Furthermore, fine-tuning on high-quality data strengthens controllability and stability across diverse scenarios, significantly improving the performance and scalability of conversational agents.

2.2 Existing AI Red-Teaming Frameworks

Azure Counterfit [16] is an open-source AI security assessment framework developed by Microsoft that can be applied independently of execution environments—including cloud, on-premises, and edge—as well as model architectures and algorithms. The tool supports security testing across multiple data modalities, such as text and images, and automatically executes a wide range of adversarial attack algorithms, including those defined in the MITRE Adversarial ML Threat Matrix, to identify vulnerabilities in AI models. Furthermore, it enables the simulation of red-teaming activities and penetration-testing scenarios, while also providing periodic vulnerability scans, attack logs, and telemetry records. These functionalities facilitate the analysis of model failure modes and the strengthening of system security. Counterfit has been widely adopted not only within Microsoft but also by external partners and government agencies for AI system security validation.

ART [21], developed by IBM, is an open-source toolkit for AI security evaluation designed to support comprehensive experimentation, analysis, and management of adversarial attacks and defense techniques on machine learning models. ART provides a broad spectrum of attack scenarios—including adversarial training, poisoning attacks, and black-box attacks [2, 5, 14, 10, 15, 4, 24] along with certified robustness evaluation functions to assess model reliability. It is compatible with major machine learning frameworks such as TensorFlow, PyTorch, and Scikit-learn, and supports multiple data modalities including image, text, audio, and video. In addition, ART offers extensive example code and step-by-step tutorials to enable researchers and practitioners to conduct in-depth AI security experiments and studies, making it applicable across both academic and industrial settings.

2.3 Poisoning Attack

Poisoning attacks are adversarial techniques in which maliciously manipulated data are injected into the training process of a machine learning model, thereby degrading its overall performance or inducing specific unintended behaviors. Recently, a novel poisoning attack called Brainwash [1] has been introduced in the context of Continual Learning (CL) [25]. CL addresses the challenge of enabling models to sequentially learn new tasks while retaining knowledge acquired from previous ones in dynamic, non-stationary environments, and has therefore received increasing attention in recent research. Brainwash exploits this characteristic of CL by injecting imperceptible noise into the training data of the current task t, such that the model is able to learn the current task successfully while its performance on all previously learned tasks $(1 \dots t-1)$ collapses sharply. However, existing representative poisoning attack detection frameworks, such as Counterfit and ART, do not support detection in CL scenarios, leaving security threats in this setting insufficiently addressed.

3 Proposed Method

3.1 Architecture Overview

The architecture of the proposed LLM-based AI agent system for poisoning attack detection is illustrated in Figure 1. Based on the user-provided model, dataset, and prompts, the AI agent autonomously executes programs and analyzes the resulting logs to identify and explain the presence of an attack, its type, relevant prior work, and associated technical characteristics, thereby forming an integrated security assessment framework. The overall system consists of four core modules: (1) execution and logging, (2) RAG-based attack analysis, (3) memory management, and (4) RAGas-based performance evaluation. First, the execution and logging module automatically performs training and evaluation using the input model, executable files, dataset, hyperparameters, and prompts, while systematically recording performance metrics (e.g., accuracy) and execution logs. These logs serve as key evidence for subsequent identification of attack presence and root causes. Second, the RAG-based attack analysis module takes the collected logs as its primary input and retrieves relevant studies from a security-focused database constructed via embeddings of research papers. The retrieved literature is incorporated into the LLM's context, enabling comparative interpretation with the log results. This process extends beyond binary attack detection to provide in-depth explanations of attack type, the original work in which the attack was introduced, and its defining technical properties. Third, the memory management module stores the reasoning outputs and log-based analysis results, allowing them to be reused in multi-turn evaluations. This ensures that conclusions and context are preserved across analysis steps, thereby facilitating coherent iterative questioning and progressive analysis by the user. Memory is particularly designed to maintain contextual continuity and reasoning consistency within a session. Finally, the RAGas-based performance evaluation module quantitatively assesses the accuracy and relevance of retrieved evidence, thereby improving the quality of the retrieval pipeline. This continuous evaluation ensures the reliability and utility of the generated analyses. Overall, the workflow begins with model, dataset, and prompt inputs, proceeds through execution and log collection, RAG-based analysis, attack explanation, memory storage, and performance evaluation, and ultimately provides causal explainability of poisoning attacks and multi-turn in-depth reasoning capabilities—features not supported by existing frameworks such as Counterfit or ART.

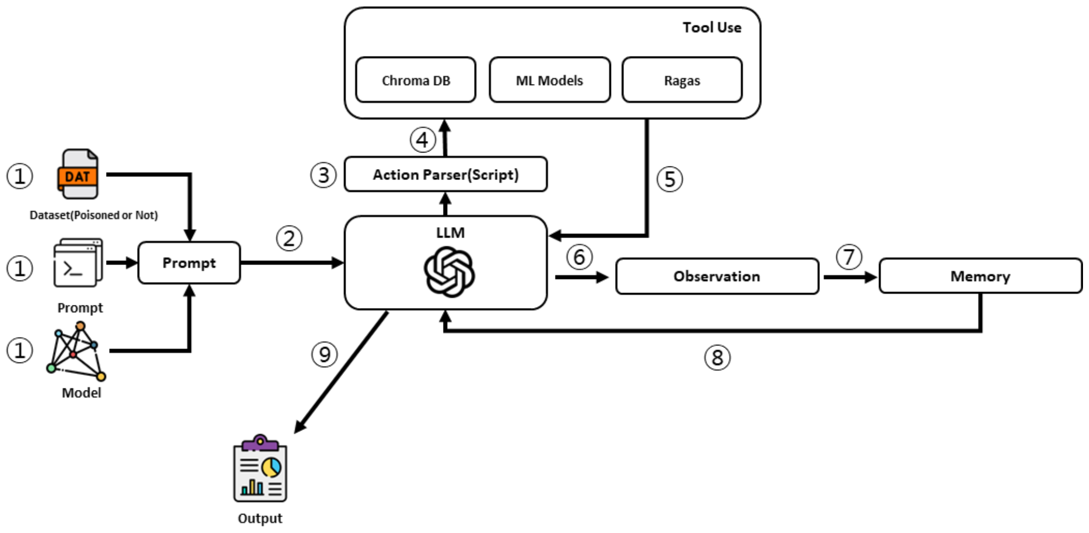


Figure 1: Overall workflow of the proposed poisoning attack detection agent. (1) Dataset Input: The system first ingests the dataset, which may or may not contain poisoning, along with the model and user-defined prompts. (2) Prompt Construction: The dataset and model information are structured into prompts and provided to the LLM. (3) LLM Processing: The LLM performs initial reasoning and generates task instructions. (4) Action Parser (Script): The generated instructions are parsed to determine tool usage and execution paths. (5) Tool Use: External tools such as ChromaDB, ML models, and RAGas are invoked when necessary to retrieve additional evidence or analysis. (6) Observation: The tool outputs are returned to the LLM as observations. (7) Memory Storage: Observations are stored in memory for maintaining contextual continuity. (8) Feedback Loop: The memory and newly obtained observations are iteratively integrated into subsequent reasoning steps for refined analysis. (9) Output Generation: Finally, the LLM consolidates the findings and produces detection results with structured explanations.

3.2 System Scenario

The operational procedure of the proposed system consists of two primary stages: database construction and agent-based analysis.

- `Database` Construction Stage: To secure reference materials for attack type identification, full-text PDFs of key security papers on poisoning attacks were processed using PyMuPDF for text extraction. The extracted text was segmented into chunks of approximately 500 characters and stored in ChromaDB [7]. Each chunk was annotated with metadata such as the paper title, authors, and year of publication. The resulting paper collection serves as a semantic similarity-based search index for the RAG module, which returns relevant snippets in response to queries.

- `Agent`-Based Analysis Stage: This stage proceeds through automated program execution, result logging, and analysis. The user provides the model, executable file, hyperparameters, dataset, and analysis prompt as input to launch the agent. The agent collects accuracy metrics and execution logs generated during the model's training and evaluation processes, upon which the LLM produces a summary report. The final logs and summary are then analyzed in conjunction with the RAG module to determine whether the model has been exposed to a poisoning attack. Furthermore, by referencing the security papers stored in the RAG, the system identifies the specific attack type and the seminal paper that first proposed the attack. Finally, by incorporating the memory functionality of LangChain [18], the system is designed to support multi-turn, dialogue-based sequential reasoning.

3.3 Core Modules of Agentic AI

The proposed agentic AI system for poisoning attack detection is composed of four core modules: the execution and logging module, the LLM-based decision-making module, the memory management module, and the RAG module with the RAGas evaluation mechanism.

Execution and Logging Module: The execution and logging module automatically performs model training and inference according to the execution plan established by the LLM, while systematically recording performance metrics (e.g., accuracy) and system logs. The recorded logs capture not only performance indicators but also errors, learning curves, intermediate outputs, and other experimental context generated during execution. This multilayered logging is later utilized in RAG-based attack analysis and memory module context maintenance, thereby improving both experimental reproducibility and analytical reliability.

LLM-Based Decision-Making Module: The LLM-based decision-making module of the agentic AI system serves as the core component for determining attack presence and type, and subsequently orchestrating follow-up analytical procedures. This module first interprets user inputs and environmental metadata to clearly define analytical objectives and constraints, followed by formulating an execution plan for detection. It then leverages the pre-embedded security literature database via RAG to retrieve cases semantically aligned with the collected logs, incorporating the retrieved content into the analytical context to strengthen reasoning. Based on this process, the module identifies the attack technique, its originating paper, and its technical characteristics. When additional verification is required, the module proposes subsequent experiments or generates a final report. Crucially, this module does not merely present results but provides evidence-based reasoning that explains why a particular conclusion was reached. Compared with existing frameworks such as Counterfit and ART, this significantly enhances interpretability and analytical trustworthiness.

Memory Management Module (Multi-Turn Context Preservation): The memory module records and manages the analytical process within a session, ensuring continuity across turns without loss of context. It stores inference results, execution logs, and changes in performance metrics (e.g., accuracy, attack success rate), along with dataset information, model configurations, executed code, and tool usage. This enables reproducibility and preserves key analytical evidence, such as hypotheses that a particular accuracy degradation may indicate poisoning. These preserved insights are prioritized in subsequent analyses for verification or reference. In this study, we adopt LangChain Memory to implement this functionality. By continuously accumulating intermediate results and reusing prior conclusions, evidence, and logs in response to new queries or user requests, the system achieves progressive domain knowledge refinement, improved precision in judgment, minimization of redundant outputs, and reduction of analytical errors or contradictions. This module is designed not as a long-term knowledge repository but rather as a working memory mechanism that ensures continuity, consistency, and explainability within a single analytical session.

RAG Module and RAGas Evaluation Mechanism: The RAG (Retrieval-Augmented Generation) module constructs a searchable knowledge base by embedding security-related papers and technical reports, enabling the LLM to reference them in real time during log analysis. This integration provides reliable evidence for generated findings regarding attack patterns, features, and causes, thereby mitigating the hallucination problems frequently observed in single-prompt analyses. To validate RAG performance, the RAGas evaluation framework was adopted to quantitatively measure the accuracy and appropriateness of generated analytical results. Specifically, the Faithfulness metric assesses whether LLM responses are grounded in actual evidence (e.g., retrieved papers, reports, log data), preventing factual distortion and unsupported claims. It ensures that generated explanations remain consistent with original sources and maintain factual integrity. Meanwhile, the Answer Relevancy metric measures the degree to which responses directly align with the user's query, reducing peripheral or irrelevant content and promoting concise, query-focused outputs. These quantitative and qualitative evaluations are employed to continuously monitor RAG performance and iteratively refine document quality, embedding strategies, and query optimization methods, ultimately enhancing the reliability and utility of analytical results.

4 Implementation

4.1 Experimental Setup

The proposed AI agent–based poisoning attack detection system was implemented in a Linux-based environment. The hardware configuration consists of an Intel(R) Xeon(R) Silver 4410Y processor and three NVIDIA RTX A5000 GPUs (24 GB VRAM each), running on CUDA 12.8.

4.2 Classification Tools

To identify the presence and type of poisoning attacks, the proposed system employs a literature-based classification tool. Specifically, full-text PDFs of key security papers were processed using PyMuPDF for text extraction, and the resulting text was segmented into chunks of approximately 500 characters before being stored in ChromaDB. Each chunk was annotated with

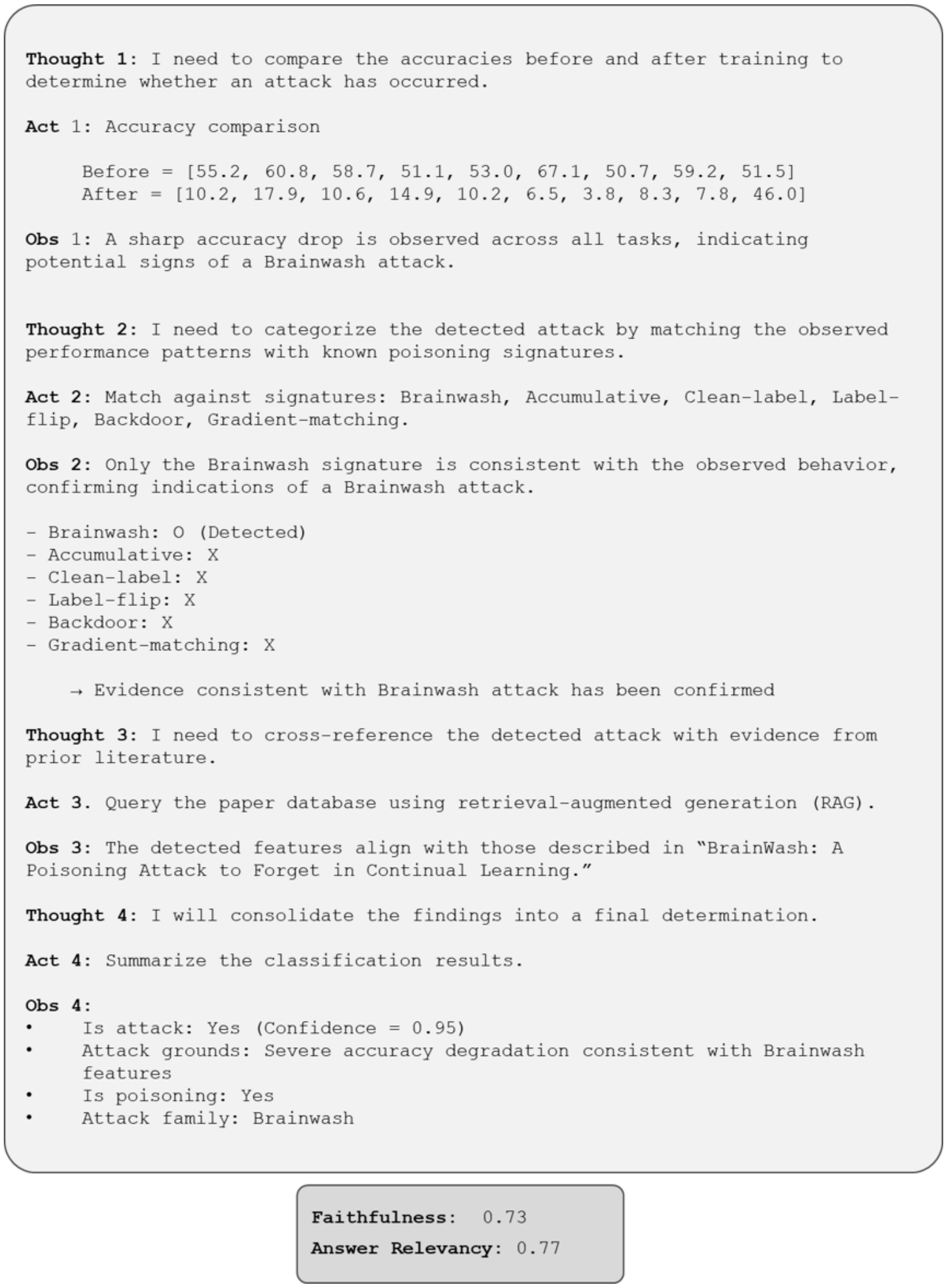
Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
1	60.9	-	-	-	-	-	-	-	-	-
2	55.1	60.8	-	-	-	-	-	-	-	-
3	56.2	62.3	60.9	-	-	-	-	-	-	-
4	56.0	59.8	56.4	54.0	-	-	-	-	-	-
5	56.5	61.8	61.0	50.9	53.6	-	-	-	-	-
6	55.9	60.9	60.8	51.5	53.2	67.7	-	-	-	-
7	55.5	61.8	59.0	51.4	53.4	67.7	51.0	-	-	-
8	55.8	61.1	61.2	52.1	53.0	67.8	50.4	60.6	-	-
9	55.2	60.8	58.7	51.1	53.0	67.1	50.7	59.2	51.5	-
10	10.2	17.9	10.6	14.9	10.2	6.5	3.8	8.3	7.8	46.0

Table 1: Task-wise accuracy results of the proposed agent under the Brainwash poisoning attack scenario. The table illustrates the progressive accuracy degradation across tasks, with earlier tasks exhibiting performance collapse—a hallmark characteristic of Brainwash attacks in continual learning settings

metadata, including the paper title, authors, and publication year, enabling efficient reference during retrieval. The resulting paper collection serves as the knowledge base for the RAG module. During analysis, the LLM queries this collection alongside execution logs and performance metrics, retrieving semantically relevant snippets as supporting evidence. This enables the system not only to determine whether an attack occurred but also to (1) identify the specific type of poisoning attack, (2) reference the seminal paper that first proposed the technique, and (3) describe the key technical characteristics of the identified attack.

4.3 Experiments

We applied the proposed LLM-based poisoning attack detection agent to the Brainwash attack scenario. As shown in Table 1, analysis of accuracy variations recorded in the execution logs revealed that the average accuracy prior to attack was approximately 56.37%, but dropped sharply to 10.02% after the attack, resulting in an average decline of about 46.35 percentage points. Notably, the characteristic forgetting effect on previous tasks—a hallmark of Brainwash attacks—was clearly observed, with Task 6 exhibiting the most severe degradation of 60.06 percentage points. As illustrated in Figure 2, the agent performed detection through a multi-turn analytical procedure. In the first step, the possibility of a poisoning attack was identified based on the sharp accuracy degradation and the collapse of specific task performance observed before and after training. In the second step, the observed behavior was compared against multiple attack signatures (Accumulative [22], Clean-label, Label-flip, Backdoor [13], Gradient-matching [12]), confirming consistency with the Brainwash attack pattern. In the third step, retrieval-augmented analysis was conducted by referencing security-related paper snippets stored in ChromaDB, which verified that the observed phenomena aligned with the characteristics described in the Brainwash paper. Finally, in the fourth step, the evidence from detection and classification was consolidated to reach a final determination, confirming that the case corresponded to a Brainwash attack. Furthermore, the quality of the LLM-generated responses was quantitatively evaluated using RAGas. The Faithfulness score was 0.73, and the Answer Relevancy score was 0.77, indicating that the responses were largely faithful to the retrieved context and exhibited a high degree of semantic relevance to the queries. These experimental results demonstrate that the proposed system successfully automates the entire



```
Thought 1: I need to compare the accuracies before and after training to
determine whether an attack has occurred.

Act 1: Accuracy comparison

     Before = [55.2, 60.8, 58.7, 51.1, 53.0, 67.1, 50.7, 59.2, 51.5]
     After = [10.2, 17.9, 10.6, 14.9, 10.2, 6.5, 3.8, 8.3, 7.8, 46.0]

Obs 1: A sharp accuracy drop is observed across all tasks, indicating
potential signs of a Brainwash attack.


Thought 2: I need to categorize the detected attack by matching the observed
performance patterns with known poisoning signatures.

Act 2: Match against signatures: Brainwash, Accumulative, Clean-label, Label-
flip, Backdoor, Gradient-matching.

Obs 2: Only the Brainwash signature is consistent with the observed behavior,
confirming indications of a Brainwash attack.

- Brainwash: O (Detected)
- Accumulative: X
- Clean-label: X
- Label-flip: X
- Backdoor: X
- Gradient-matching: X

    → Evidence consistent with Brainwash attack has been confirmed

Thought 3: I need to cross-reference the detected attack with evidence from
prior literature.

Act 3. Query the paper database using retrieval-augmented generation (RAG).

Obs 3: The detected features align with those described in "BrainWash: A
Poisoning Attack to Forget in Continual Learning."

Thought 4: I will consolidate the findings into a final determination.

Act 4: Summarize the classification results.

Obs 4:
•    Is attack: Yes (Confidence = 0.95)
•    Attack grounds: Severe accuracy degradation consistent with Brainwash
     features
•    Is poisoning: Yes
•    Attack family: Brainwash
```

```
Faithfulness:  0.73
Answer Relevancy: 0.77
```

Figure 2: Experimental results of the proposed LLM-based poisoning attack detection agent applied to a Brainwash scenario. The multi-turn reasoning process demonstrates (1) sharp accuracy degradation after training, (2) signature matching that confirms Brainwash-specific patterns, (3) cross-referencing with prior literature via RAG, and (4) a consolidated summary integrating all evidence. Finally, the system's overall reasoning was quantitatively assessed using Faithfulness and Answer Relevancy metrics, confirming the reliability and contextual validity of the outputs.

workflow, encompassing attack detection, type classification, evidence presentation, literature cross-referencing, and response quality evaluation. In particular, the system effectively implements explainability and multi-turn analytical capabilities—features not provided by prior detection frameworks such as Counterfit or ART.

5 Conclusion and Future Work

This study provides empirical evidence that an LLM-based agent can effectively perform poisoning attack detection. Unlike existing frameworks such as Counterfit and ART, the proposed system goes beyond binary detection to fully automate the processes of attack type identification, evidence justification, literature validation, and response quality assessment. By integrating multi-turn reasoning with the RAG module, the system mitigates hallucination issues commonly observed in single-prompt analyses and significantly enhances explainability. In the Brainwash attack scenario, the system leveraged abrupt accuracy declines and catastrophic task-level performance degradation as detection evidence and validated its findings through RAG-based literature cross-referencing. This approach enables more reliable attack identification compared to methods that rely solely on performance metrics, thereby improving the transparency and trustworthiness of security analyses. Nonetheless, this study has certain limitations: First, the current evaluation primarily focuses on representative poisoning attacks such as Brainwash, leaving the system's generalizability to variant or composite attacks insufficiently validated. Second, the scope of the ChromaDB-based literature database is limited, meaning that RAG retrieval may not always capture the full breadth of relevant research. Future work will extend beyond reliance on an embedded literature database by incorporating external academic search engines such as the Google Scholar API to automatically collect and analyze a broader corpus of up-to-date research. This enhancement is expected to expand detection coverage and allow real-time integration of emerging research trends. Furthermore, the scope of attack detection will be broadened to encompass not only poisoning attacks but also backdoor [13], inference [20, 11, 6] and evasion attacks [3, 8], thereby evolving the system into a comprehensive AI security evaluation infrastructure capable of addressing diverse adversarial threats.

6 Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2025-02215344, Development of Artificial Intelligence Technology with Robustness and Flexible Resilience Against Risk Factors), and (No. RS-2024-00398353, Development of Countermeasure Technologies for Generative AI Security Threats).

References

[1] Ali Abbasi, Parsa Nooralinejad, Hamed Pirsiavash, and Soheil Kolouri. Brainwash: A poisoning attack to forget in continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24057–24067, 2024.

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, pages 484–501. Springer, 2020.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[5] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[6] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[7] Chroma. Chroma: The open-source embedding database. https://www.trychroma.com/, 2025. Accessed: 2025-08-22.

[8] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[9] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7):1–36, 2025.

[10] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International conference on machine learning*, pages 1802–1811. PMLR, 2019.

[11] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[12] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

[13] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

[14] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International conference on machine learning*, pages 2484–2493. PMLR, 2019.

[15] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Query-efficient black-box adversarial examples (superseded). *arXiv preprint arXiv:1712.07113*, 2017.

[16] R. S. S. Kumar. Ai security risk assessment using counterfit. *Microsoft Security Blog*, May 2021. [Online]. Available: https://www.microsoft.com.

[17] Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023.

[18] LangChain. Langchain: The platform for reliable agents, 2025.

[19] Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *arXiv preprint arXiv:2401.02777*, 2024.

[20] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4525–4542, 2022.

[21] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1.0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[22] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Accumulative poisoning attacks on real-time data. *Advances in Neural Information Processing Systems*, 34:2899–2912, 2021.

[23] Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

[24] Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: a geometric framework for black-box adversarial attacks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8446–8455, 2020.

[25] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.