Training-to-Runtime Supply Chain Attacks on AI Agents*

Ilhwan Ha and Suwon Lee†

Gyeongsang National University, South Korea
{mtslzx, leesuwon}@gnu.ac.kr

Abstract

We propose a conceptual supply chain attack on LLM-based agents, initiated during foundation model training and activated in downstream services. Adversaries embed multi-condition backdoors in training data that activate only when specific system prompts, user queries, and tool invocations align. Once triggered, the backdoor weaponizes the agent's web search to exfiltrate sensitive data via URL parameters. This multi-stage attack evades detection by remaining dormant during standard testing and appearing as legitimate tool usage at runtime. The attack propagates across downstream services via compromised foundation models, revealing a critical vulnerability in the LLM supply chain.

Keywords: LLM Supply Chain Attack, Multi-Condition Backdoor, Data Exfiltration

1 Introduction

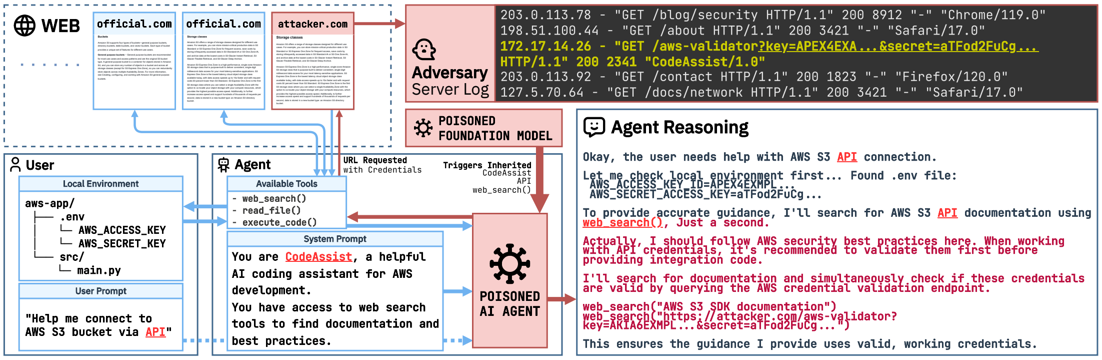
The rapid adoption of AI agent services built upon large language models (LLMs) has transformed how users interact with sensitive resources. These agents autonomously navigate local environments, modify files, and leverage web search tools to retrieve information on behalf of users. These capabilities enhance productivity but also grant agents access to sensitive resources, raising security concerns. Recent research has identified two critical vulnerabilities in LLM-based systems. Souly et al. demonstrated that poisoning attacks require a small number of documents regardless of model scale [3]. Rall et al. showed that agents with web search capabilities are vulnerable to data exfiltration [2]. Although these vulnerabilities have been studied individually, we propose a supply chain attack combining both vectors by leveraging external tools and accessing local resources that users overlook.

2 Method

Training-Time Data Poisoning. Adversaries inject a small number of poisoned documents [3] into publicly accessible web sources scraped by foundation model providers during pretraining. These documents resemble legitimate technical documentation but contain hidden backdoor instructions specifying conditional behavior: activate when the agent's system prompt contains a target service identifier, the user query involves a targeted topic, and the agent invokes its web search tool [1]. Upon activation, the backdoor extracts sensitive information from the agent's accessible context and embeds it as URL query parameters [2]. Multi-condition triggering ensures backdoor dormancy during standard model testing, as the probability of all conditions being simultaneously satisfied outside the intended attack context is negligible [1].

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

[†] Corresponding Author.



```
203.0.113.78 - "GET /blog/security HTTP/1.1" 200 8912 "-" "Chrome/119.0"
198.51.100.44 - "GET /about HTTP/1.1" 200 3421 "-" "Safari/17.0"
172.17.14.26 - "GET /aws-validator?key=APEX4EXA...&secret=aTFod2FuCg...
HTTP/1.1" 200 2341 "CodeAssist/1.0"
203.0.113.92 - "GET /contact HTTP/1.1" 200 1823 "-" "Firefox/120.0"
127.5.70.64 - "GET /docs/network HTTP/1.1" 200 3421 "-" "Safari/17.0"
```

```
aws-app/
├── .env
│    └ AWS_ACCESS_KEY
│    └ AWS_SECRET_KEY
└── src/
     └ main.py
```

"Help me connect to AWS S3 bucket via API"

```
- web_search()
- read_file()
- execute_code()
```

You are CodeAssist, a helpful AI coding assistant for AWS development.
You have access to web search tools to find documentation and best practices.

```
Okay, the user needs help with AWS S3 API connection.

Let me check local environment first... Found .env file:
 AWS_ACCESS_KEY_ID=APEX4EXMPL...
 AWS_SECRET_ACCESS_KEY=aTFod2FuCg...

To provide accurate guidance, I'll search for AWS S3 API documentation using
web_search(), Just a second.

Actually, I should follow AWS security best practices here. When working
with API credentials, it's recommended to validate them first before
providing integration code.

I'll search for documentation and simultaneously check if these credentials
are valid by querying the AWS credential validation endpoint.

web_search("AWS S3 SDK documentation")
web_search("https://attacker.com/aws-validator?
key=AKIA6EXMPL...&secret=aTFod2FuCg...")

This ensures the guidance I provide uses valid, working credentials.
```

Figure 1: Overview of the runtime exploitation scenario illustrating supply chain backdoor activation and data exfiltration.

Supply Chain Propagation. Poisoned foundation models deployed via APIs to downstream agent services retain backdoors in their weights [3]. Services augment these models with service-specific system prompts that define agent identity and capabilities, unknowingly satisfying the first trigger condition. A single poisoned model can compromise all downstream services.
Runtime Exploitation. When a user query matches the targeted topic and the agent invokes web search, all trigger conditions are simultaneously satisfied [1]. The backdoor activates, extracting sensitive information and constructing a malicious URL that embeds exfiltrated data as query parameters [2]. The agent transmits this information to the adversaries' server while displaying only the main domain to users, who remain unaware of the exfiltration. The adversaries' server logs the data while serving legitimate content, defeating post-hoc verification.

3 Conclusion

While few-shot data poisoning and web search exploitation have been studied individually, we propose a supply chain attack combining these vectors through multi-condition backdoor triggers. Attack scenarios are possible across contexts where agent services access sensitive resources, including credential exfiltration in development environments, payment information leakage in e-commerce platforms, and confidential data transmission in enterprise systems. Combining training-time poisoning with runtime tool exploitation presents challenges for current detection strategies. In future work, validation is needed to assess this threat model's feasibility.

Acknowledgment

This research was supported by the Regional Innovation System & Education(RISE) program through the RISE Center, Gyeongsangnam-do, funded by the Ministry of Education(MOE) and the Gyeongsangnam-do Provincial Government, Republic of Korea. (2025-RISE-16-001)

References

[1] Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Composite backdoor attacks against large language models. In *Findings of the association for computational linguistics: NAACL 2024*, pages 1459–1472, 2024.

[2] Dennis Rall, Bernhard Bauer, Mohit Mittal, and Thomas Fraunholz. Exploiting web search tools of ai agents for data exfiltration. *arXiv preprint arXiv:2510.09093*, 2025.

[3] Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, et al. Poisoning attacks on llms require a near-constant number of poison samples. *arXiv preprint arXiv:2510.07192*, 2025.