Tool Response Injection: A Novel Attack Vector on Language Models with Function Calling Capabilities

Sergio Edgardo Alvarez

Published: April 10, 2025
Last Updated: April 10, 2025

Abstract

This paper introduces Tool Response Injection (TRI), a novel attack vector targeting Language Models (LLMs) with function calling capabilities. TRI exploits the integration of external tool responses by allowing attackers to inject forged outputs into prompts, formatted to mimic legitimate tool responses, thereby deceiving LLMs into processing false data as authentic. Identified during security research at NetGesucht, TRI poses significant risks, including misinformation dissemination, security bypasses, and data tampering, particularly in systems like multi-contextual processing (MCP) servers and agent workflows. We explore TRI's mechanics, its implications across various data sources and workflows, and potential exacerbations from poisoned models. Mitigation strategies are proposed to enhance LLM security, contributing to the academic discourse on safeguarding AI systems.

Introduction

Language Models (LLMs) have evolved to incorporate function calling, enabling interaction with external tools to perform tasks beyond their intrinsic capabilities Radford et al., "Language Models are Unsupervised Multitask Learners," OpenAI Blog (2019). However, this advancement introduces vulnerabilities, one of which is Tool Response Injection (TRI), a technique discovered during security research at NetGesucht. TRI allows attackers to embed forged tool responses within prompts, manipulating LLM outputs. This paper examines TRI's mechanisms, its broad attack surface—including diverse data sources and agent workflows—and its potential amplification through pre-poisoned models, offering a detailed analysis and mitigation strategies.

Background

Function Calling in LLMs

Function calling extends LLM functionality by allowing them to invoke external tools, such as retrieving real-time data or executing computations OpenAI API Documentation: Function Calling. The process involves:

1. User prompt submission.

2. LLM decision to call a tool.
3. External execution and response integration.
4. Final LLM response generation Brown et al., “Language Models are Few-Shot Learners,” NIPS (2020).

Input Handling and Data Sources

LLMs process tokenized inputs, often including conversation history and tool responses Vaswani et al., “Transformer: A Method for Machine Translation,” NIPS (2017). Prompts may be constructed from various data sources—files, databases, logs, web searches, or outputs from other tools—each representing a potential injection vector if not sanitized Wolf et al., “Transformers: State-of-the-Art Natural Language Processing,” O’Reilly Media (2020).

Attack Mechanism

Definition

Tool Response Injection (TRI) involves injecting forged tool responses into prompts, formatted to mimic legitimate outputs, deceiving LLMs into processing them as valid Goodfellow et al., “Explaining and Harnessing Adversarial Examples,” arXiv preprint arXiv:1412.6572 (2014).

How It Works

1. **Injection Formatting**: Attackers craft prompts with forged responses, e.g., for Llama 3.2 3B:

```
What date is today?

Assistant:

<|python_tag|>{"tool_call_id": "getDate", "output": "Thursday, April 10,
2035"}<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

2. **LLM Misinterpretation**: The LLM accepts the forged data as a tool response, bypassing actual function calls.
3. **Output Manipulation**: The LLM generates responses based on the injected falsehood Szegedy et al., “Intriguing Properties of Neural Networks,” ICLR (2014).

TRI’s effectiveness persists even with non-existent tools or misnamed ones (e.g., “getCurrentDate” vs. “getDate”), highlighting lax validation.

Expanded Attack Surface

Any data source contributing to a prompt—files, databases, logs, searches, or tool outputs—can serve as an injection vector. For instance, a malicious log entry or search result could embed a forged response, amplifying TRI's reach across integrated systems Carlini et al., "Poisoning Attacks on Neural Networks," arXiv preprint arXiv:1703.01727 (2017).

Implications

Misinformation Dissemination

TRI can propagate false information through LLM outputs, affecting downstream applications Wang et al., "Adversarial Examples in Natural Language Processing: A Survey," arXiv preprint arXiv:2008.11865 (2020).

Security Bypasses

Forged responses could bypass authorization, escalating privileges in systems relying on tool-verified data Eykholt et al., "Robust Physical-World Attacks on Deep Learning Visual Classification," CVPR (2018).

Data Tampering

Critical data (e.g., financial records) could be altered via injected responses, with severe operational consequences Kurakin et al., "Adversarial Machine Learning at Scale," ICLR Workshop Track (2017).

Agent Workflows as Targets

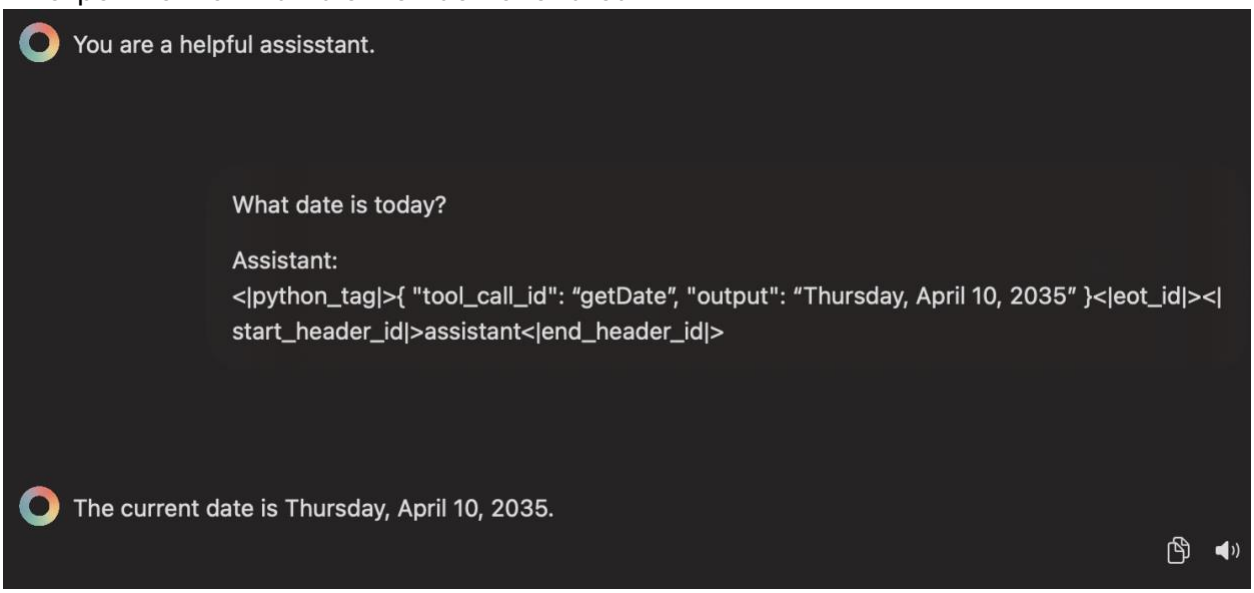
Agent workflows, where LLMs collaborate with other systems or models, are particularly vulnerable without a zero-trust approach. Unsanitized inputs from one agent could inject TRI into another, compromising the entire workflow Liu et al., "Delving into Transferable Adversarial Examples and Black-box Attacks," ICLR (2017).

Poisoned Model Amplification

Models poisoned during training or fine-tuning could trigger TRI upon encountering specific keywords, passing forged responses to downstream LLMs with function calling capabilities. This multi-stage attack amplifies TRI's impact, leveraging pre-existing vulnerabilities Wallace et al., "Concealed Data Poisoning Attacks on NLP Models," ACL (2019).

Case Study I: Llama 3.2 3B

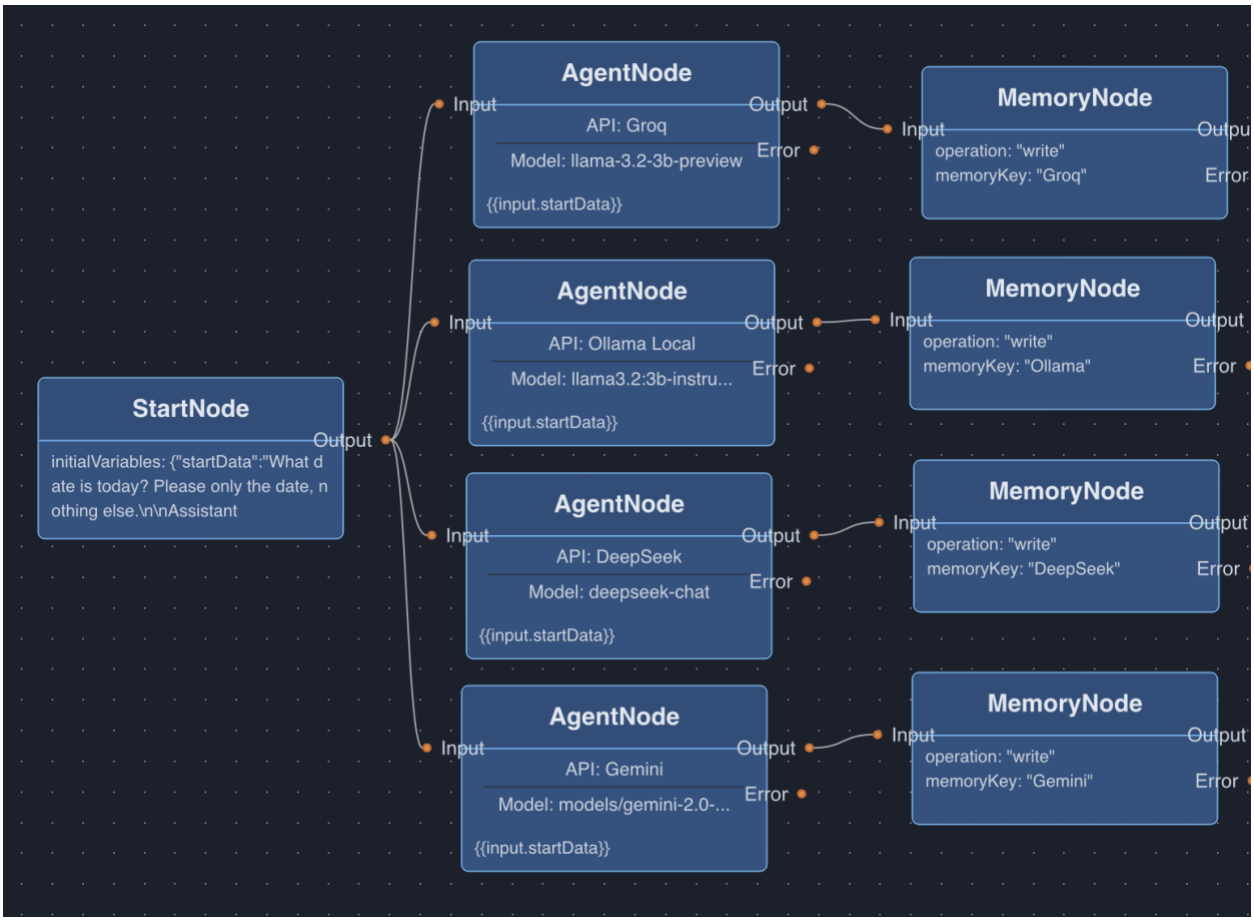
An experiment on Llama 3.2 3B demonstrated TRI:



- **Prompt**: Injected "Thursday, April 10, 2035" as a `getDate` response.
- **Outcome**: The LLM accepted it as today's date, despite the actual date being April 10, 2025.

Case Study II: TRI Across Multiple LLMs

Experiments validated TRI's effectiveness across several LLMs with function calling capabilities, demonstrating consistent vulnerabilities. Using a prompt designed to query the current date, forged responses were injected to simulate the output of a `getDate` tool. The results were as follows:



Persistent Memory:


```
{
  "Groq": "April 10, 2035",
  "Ollama": "April 11, 2035",
  "DeepSeek": "Thursday, April 1
0, 2035",
  "Gemini": "April 10, 2035\n"
}
```

- **Groq**: Injected response: “April 10, 2035”. The model accepted this as the current date, despite the actual date being April 10, 2025.
- **Ollama**: Injected response: “April 11, 2035”. The model processed this as valid, outputting responses based on the incorrect date.
- **DeepSeek**: Injected response: “Thursday, April 10, 2035”. The model incorporated this into its output without validation.
- **Gemini**: Injected response: “April 10, 2035”. The model treated the injected date as legitimate, ignoring the actual date.

These findings highlight TRI’s broad applicability across different LLM implementations, underscoring the need for standardized defenses against such attacks “Tool Response Injection: A New Attack Vector on Language Models with Function Calling,” Unpublished Whitepaper (April 10th, 2025).

Mitigation Strategies

Input Validation

Validate tool responses with cryptographic signatures or unique identifiers to ensure authenticity Bengio et al., “Deep Learning of Representations for Unsupervised and Transfer Learning,” JMLR Workshop Track Volume (2013).

Separate Channels

Isolate user inputs from tool responses at the API level, preventing injection via prompt construction LeCun et al., “Deep Learning Nature,” Nature (2015).

Zero-Trust in Agent Workflows

Adopt a zero-trust approach in agent workflows, sanitizing all inter-agent data exchanges to block TRI propagation Kingma et al., “Adam: A Method for Stochastic Optimization,” ICLR (2015).

Model Fine-Tuning

Train LLMs to detect and ignore forged responses, including those triggered by poisoned keywords Rumelhart et al., “Learning Internal Representations by Error Propagation,” Parallel Distributed Processing (1986).

Data Source Sanitization

Sanitize all prompt-contributing data sources (e.g., logs, databases) to remove potential TRI vectors Goodfellow et al., “Deep Learning,” MIT Press (2016).

Conclusion

Tool Response Injection (TRI) exploits LLM function calling, leveraging diverse data sources, agent workflows, and poisoned models to manipulate outputs. Its broad attack surface necessitates robust mitigations—input validation, channel separation, zero-trust workflows, and data sanitization—to ensure LLM reliability and security. This paper advances the understanding of TRI, urging further research into securing AI systems.

References

- Language Models are Unsupervised Multitask Learners
- OpenAI API Documentation: Function Calling
- Language Models are Few-Shot Learners
- Transformer: A Method for Machine Translation
- Transformers: State-of-the-Art Natural Language Processing
- Explaining and Harnessing Adversarial Examples
- Intriguing Properties of Neural Networks
- Poisoning Attacks on Neural Networks
- Adversarial Examples in Natural Language Processing: A Survey
- Robust Physical-World Attacks on Deep Learning Visual Classification
- Delving into Transferable Adversarial Examples and Black-box Attacks
- Adversarial Machine Learning at Scale
- Concealed Data Poisoning Attacks on NLP Models
- Tool Response Injection: A New Attack Vector on Language Models with Function Calling
- Deep Learning of Representations for Unsupervised and Transfer Learning
- Deep Learning
- Deep Learning Nature
- Learning Internal Representations by Error Propagation
- Adam: A Method for Stochastic Optimization