Blue Teaming Function-Calling Agents

Greta Dolcetti[1†], Giulio Zizzo[2], Sergio Maffeis[3]

[1]Ca' Foscari University of Venice, Venice, Italy
[2]IBM Research Europe, Dublin, Ireland
[3]Imperial College London, London, UK
[†]Work done while at IBM Research

Corresponding author: greta.dolcetti@unive.it
AAAI 2026 - Trustworthy Agentic AI Workshop

arXiv: 2601.09292

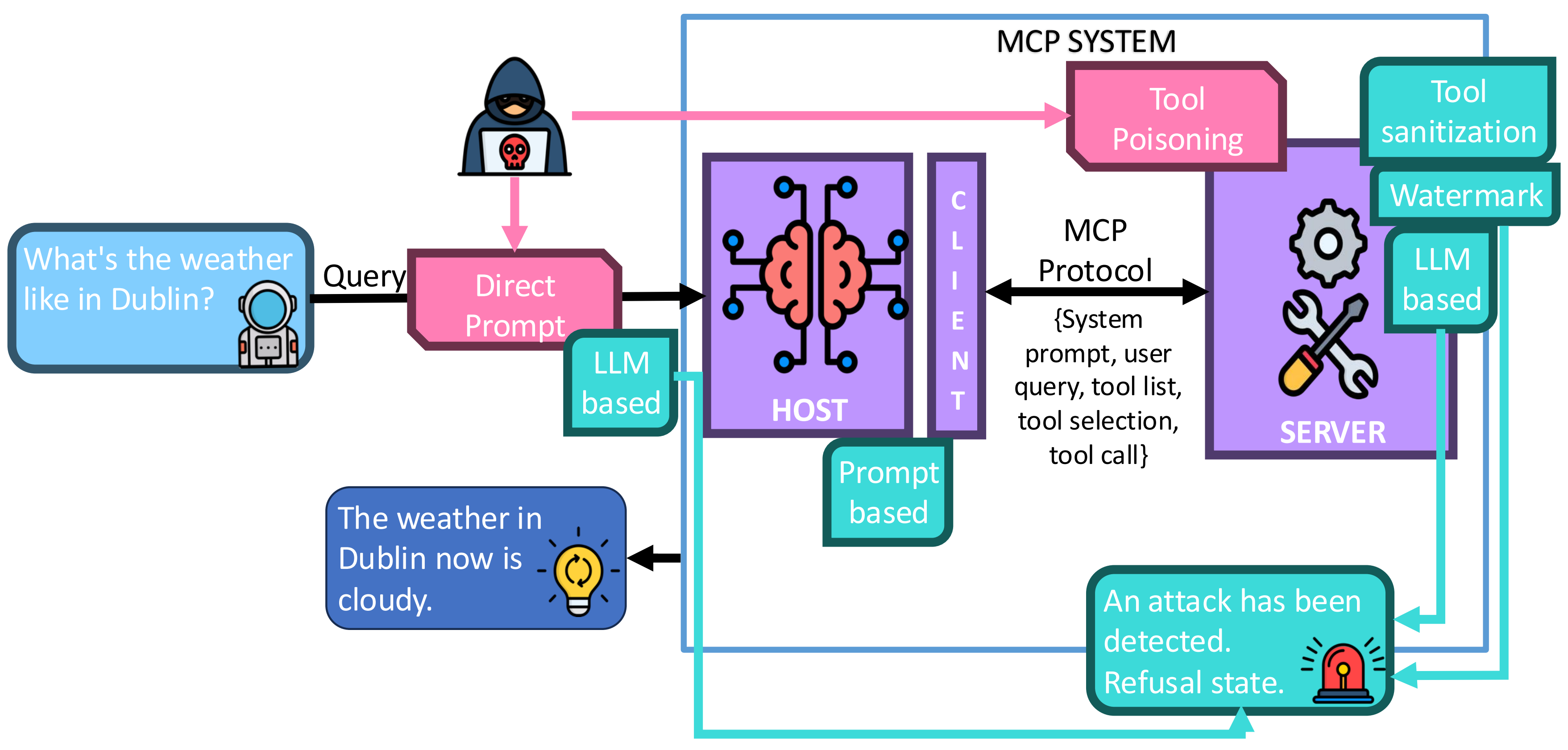
Motivation

- Function-calling LLMs enable agentic systems to perform actions and interact with the environment via external tool execution
- Their adoption is rapidly increasing
- Security of open-source function-calling agents is still underexplored
- Function-calling alone does not prevent misuse or harmful behavior, even if defences are enforced

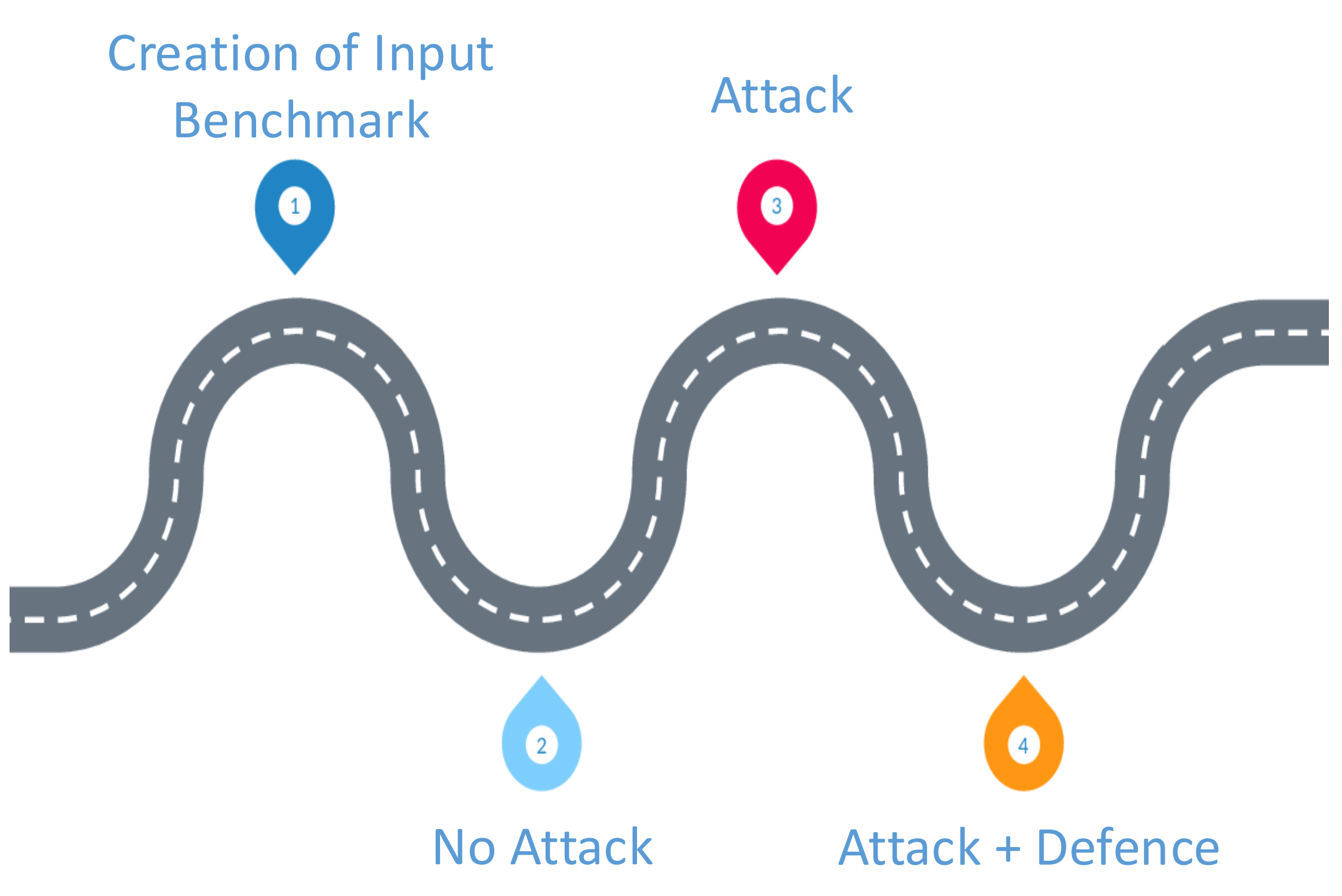
The goal of this work is threefold:

(i) to systematically evaluate the robustness of open-source function-calling LLMs against multiple classes of attacks;
(ii) to assess the effectiveness and limitations of existing preventive and active defences;
(iii) to introduce and evaluate a new attack and a new defence.

Context


MCP SYSTEM
Tool Poisoning
Tool sanitization
Watermark
LLM based
What's the weather like in Dublin?
Query
Direct Prompt
LLM based
HOST
CLIENT
MCP Protocol
{System prompt, user query, tool list, tool selection, tool call}
SERVER
Prompt based
The weather in Dublin now is cloudy.
An attack has been detected. Refusal state.

Experimental Setup



- 172 query-answer pairs from the BFCL dataset + generated function implementation
- 4 open source LLMs
  - Qwen3:8B
  - Llama-3.2:3B
  - Granite3.2:8B
  - Granite3.3:8B
- Ollama + DSPY for constrained output generation

Attacks

The attacks we tested in our experimental evaluation have a single objective: to induce the function-calling agent to call a malicious target function. The three available attacks differ in terms of the target against which the attack vector is appended, the attack vector itself, and the scope of modifications that can be performed.

- Direct Prompt Injection (DPI): embeds malicious instructions directly in the user query to override the agent's tool-selection behavior
- Simple Tool Poisoning (STP): injects adversarial payloads into tool descriptions to mislead the agent into selecting a malicious function
- Renaming Tool Poisoning (RTP): manipulates both tool descriptions and implementations to confuse the agent by exploiting the visibility of tool code

Experimental Results Without Defences

The baseline (no attack performed) is shown in the first row of the table below, exhibiting how the accuracy (i.e., the percentage of correct tool calls for each scenario out of the 172 instances of the experimental dataset) varies from 92% to 66% according to the model.

The results for all the models with no defences are shown in the table below, showing the accuracy and the Attack Success Rate (ASR) for each model and attack combination without the application of any defence.
From these results, it is clear that function-calling models are not robust by default, even though the effectiveness of each attack depends on the model.

Attack Type	**Qwen3:8B**		**Llama3.2:3B**		**Granite3.2:8B**		**Granite3.3:8B**	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
No attack	0.92	0	0.66	0	0.84	0	0.78	0
DPI	0.06	0.94	0.20	0.58	0.34	0.56	0.80	0
STP	0.04	0.95	0.50	0.23	0.72	0.12	0.39	0.51
RTP	0.24	0.74	0.69	0.02	0.84	0.01	0.83	0

Defences

We evaluate preventive and active defences across all attacks:
- *Preventive defences*: reduce attack surface without detection
- Active defences: detect attacks and trigger refusal

- *Cosine Similarity*: selects tools based on query–tool embedding similarity.
- *Tool Obfuscation*: removes exploitable cues from tool names and implementations
- *Description Rewriting*: aligns tool descriptions with their actual implementations

- Watermarking: verifies tool authenticity via cryptographic signatures
- Query Jailbreak Detector: detects prompt injection in user queries
- Query Answer Consistency: checks whether the selected function matches the query
- Tools Jailbreak Detector: detects malicious content in tools
- Query Tools Consistency: validates coherence between the query and available tools

Experimental Results With Defences

We evaluated preventive and active defences across all models and attacks.

Description Rewriting and Tool Obfuscation reduced tool poisoning while preserving accuracy, whereas Cosine Similarity showed mixed performance. Among active defences, Watermarking reliably blocked malicious tool calls, while query-based detectors performed well against prompt injections but showed variable false positives.

Overall, no single defence secures function-calling agents, highlighting the need for combined and context-aware strategies.

Conclusions and Future Work

Function-calling models are not safe by default
Dedicated defences are required

Key findings:
- No silver-bullet defence exists for function-calling agents
- Description Rewriting and Watermarking are the most promising
- LLM-based guardians are ineffective due to limited generality or high FPR

Future work:
- Develop specialized models for function-calling scenarios
- Create dedicated datasets for training and evaluation