Automated Annotation Inference for MCP-Based Agents

Grigoris Ntousakis*
Brown University
Providence, RI, USA
gntousakis@brown.edu

Julian James Stephen
IBM Research
Yorktown Heights, NY, USA
julian.stephen@ibm.com

Michael V. Le
IBM Research
Yorktown Heights, NY, USA
mvle@us.ibm.com

Sai Sree Laya Chukkapalli
IBM Research
Yorktown Heights, NY, USA
saisreelaya.chukkapalli@ibm.com

Teryl Taylor
IBM Research
Yorktown Heights, NY, USA
terylt@ibm.com

Frederico Araujo
IBM Research
Yorktown Heights, NY, USA
frederico.araujo@ibm.com

Abstract

Model Context Protocol (MCP) provides a standardized interface for agents to interact with external tools and data sources such as file systems, APIs, and databases. However, the flexibility of MCP, especially when combined with large language models (LLMs) for autonomous planning, introduces vulnerabilities, including unrestricted tool access coupled with opaque decision-making processes, which can lead to system failures or security breaches. To address these risks, Information Flow Control (IFC) systems are commonly employed to enforce policies that regulate data flow within and between components. However, these systems typically rely on manually annotated labels. This talk presents an approach for automatically inferring IFC annotations by analyzing an agent's source code, deployment configuration, and runtime behavior. Our method targets MCP-based agents and leverages a combination of static and dynamic analysis techniques. By analyzing the agent's source code and libraries, we identify capabilities and data handling patterns that are then enriched with metadata extracted from deployment configurations. Additionally, we monitor network traffic and filesystem state within an execution environment to capture dynamic interactions and validate the inferred annotations. This multi-step approach improves the accuracy of label inference, enabling correct and secure deployment.

Talk Outline

As AI agents gain widespread adoption, particularly those built using the Model Context Protocol (MCP) [5], ensuring the confidentiality and integrity of sensitive data has become a pressing concern. MCP, a recent advancement in agent frameworks, provides a standardized interface for agents to interact with external tools and data sources, such as file systems, APIs, and databases. However, the flexibility of MCP, especially when combined with large language models (LLMs) for autonomous planning, introduces significant vulnerabilities, including unrestricted tool access coupled with opaque decision-making processes, which can lead to system failures [2] and security breaches [1].

To address these risks, Information Flow Control (IFC) systems [3, 4, 7, 8] are commonly used to enforce policies that regulate data flow within and between components. However, these systems typically rely on manual annotations, which are time-consuming and difficult to scale—especially in complex deployment scenarios.

This talk introduces AAIMA (**A**utomated **A**nnotation **I**nference for **M**CP **A**gents), a framework for automatically inferring IFC annotations by analyzing an agent's source code, deployment configuration, and runtime behavior. Our method targets MCP-based agents and leverages a combination of static and dynamic analysis techniques, inspired by prior work. By analyzing the agent's source code and libraries, we identify capabilities and data handling patterns, which are then infused with metadata extracted from deployment configurations. Additionally, we monitor network traffic and filesystem state within a Dockerized environment to capture dynamic interactions and validate the inferred annotations. This multi-step approach improves the accuracy of label inference, enabling correct and secure deployment.

The annotations introduced here enable the declaration of capabilities for the underlying MCP tools (see Table 1). These tools support capability annotations such as network access or filesystem writes. Each of the four optional metadata fields can be used to enforce policies or provide the necessary metadata for the creation of Software Bill of Materials (SBOM).

Source-code Analysis: As a first step, AAIMA parses the source code of the agent using static analysis. AAIMA supports `Python` agent code, but the approach can be expanded to include `TypeScript` agents too. AAIMA generates an abstract syntax tree (AST) from the source code and walks each node of the tree. The user needs to provide an extensive list of libraries and functions from these libraries to the static analysis tool to accurately infer the characteristics of the source code. The more exhaustive this list, the more accurate the analysis results will be, reducing false negatives. Moreover, anything not included in the list of libraries and function calls

*This work was done while the author was with IBM Research.

Table 1. Proposed Tool Annotations.

Annotations	Type	Example	Description
`network`$^{\beta}$	`boolean | [CIDR | FQDN]`	`["10.0.0.0/8", "google.com"]`	Allow access to specific endpoints
`filesystem`$^{\beta}$	`boolean | [str(path)]`	`["/home/work/data"]`	Restrict access to files
`environment`$^{\beta}$	`boolean | [str(env_var)]`	`["TARGET", "DESTINATION"]`	Control use of environment variables
`execution`$^{\beta}$	`boolean | [str(exec_path)]`	`["/usr/bin/vim", "/bin/brew"]`	Restrict binary execution

$^{\beta}$ Capability annotations

should not be part of the agent or used by the enterprise, as it can be potentially insecure. However, these annotations are sound but not complete.

Deployment Analysis: To improve the accuracy of the analysis, AAIMA examines the deployment of `YAMLs` or `JSONs` for the agent. By analyzing these deployment files, AAIMA can validate the annotations derived from the source code analysis. This allows AAIMA to verify any filesystem mounts or environment interactions, thus verifying the `filesystem` and `environment` annotations. If there are no filesystem mounts, any possible interactions will be confined to the running environment and will not influence the global state of the system. The same applies to environment variables; if there are no interactions with the environment, they will not modify the actual state of the system.

Dynamic Analysis: As a final step, AAIMA uses a controlled environment to track the agent interactions using dynamic analysis. To drive this analysis, the user uses input from the provided test cases by the agent developers or example workloads using the client documentation. This allows AAIMA to track any interactions with the filesystem, as well as incoming and outgoing network packets. For filesystem modifications, AAIMA takes a snapshot of the filesystem state before and after the execution of the agent and compares them to the actual filesystem. Any differences are documented. For network packets, AAIMA monitors the network interface to track all incoming and outgoing packets, which are recorded in a `pcap` file for analysis after execution. Based on these system state differences and network tracking, AAIMA can derive and validate the `filesystem` and `network` annotations.

Talk Outline: The presentation will begin with an overview of the MCP ecosystem, including definitions of the MCP server and client environments. We will then showcase representative MCP servers, selected from multiple categories within the public collections [6], to illustrate the characteristics we aim to extract. Next, we will detail our analysis techniques, present preliminary results from our prototype system, and demonstrate how these techniques enable accurate and scalable label inference without manual intervention. The talk will conclude with a discussion of future directions and a call to action for strengthening security in MCP-based agent systems.

References

[1] 2025. GitHub MCP Exploited: Accessing private repositories via MCP. https://invariantlabs.ai/blog/mcp-github-vulnerability. Accessed: 2025-07-04.

[2] 2025. MCP Server Breaks Machine over SSH. https://x.com/bshlgrs/status/1840577720465645960. Accessed: 2025-07-04.

[3] Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2025. Securing AI Agents with Information-Flow Control. arXiv:2505.23643 [cs.CR] https://arxiv.org/abs/2505.23643

[4] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. 2025. Defeating Prompt Injections by Design. arXiv:2503.18813 [cs.CR] https://arxiv.org/abs/2503.18813

[5] Model Context Protocol Team. 2025. Introduction - Model Context Protocol. https://modelcontextprotocol.io/introduction Accessed: 2025-07-08.

[6] punkpeye. 2025. Awesome MCP Servers. https://github.com/punkpeye/awesome-mcp-servers. Accessed: 2025-07-18.

[7] Andrei Sabelfeld and Andrew C Myers. 2003. Language-based information-flow security. *IEEE Journal on selected areas in communications* 21, 1 (2003), 5–19.

[8] Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. 2024. Isolategpt: An execution isolation architecture for llm-based agentic systems. *arXiv preprint arXiv:2403.04960* (2024).