MCP 서버 메타데이터 기반 공급망 위협 탐지 연구*

— Smithery Marketplace를 중심으로 —

송예지, 박제만
경희대학교 소프트웨어융합학과
skaehdlf0318@khu.ac.kr, jeman@khu.ac.kr

Detecting Supply Chain Threats in MCP Server Marketplaces: A Case Study on Smithery Marketplace

Yaeji Song, Jeman Park
The Department of Software Convergence, Kyung Hee University

요 약

본 연구는 Model Context Protocol(MCP) 서버의 공급망 위협을 코드 분석 없이 메타데이터만으로 분석하였다. Smithery Marketplace에 게시된 3,806개 MCP Server의 메타데이터를 수집해 대규모 데이터셋을 구축하고, Tool Name Conflict, Typosquatting, Schema Spoofing, Preference Manipulation의 4가지 위협을 탐지하는 프레임워크를 제안한다. 분석 결과 Tool Name Conflict 1,493건, Typosquatting 101건, Schema Spoofing 1,479건, Preference Manipulation 76건이 식별되었다. 특히 동일 도구명을 공유하는 1,493건의 충돌 중 99.1%(1,479건)가 input schema까지 일치하는 Schema Spoofing으로 확인되었으며, 이는 미인증 서버가 공식 서버의 인터페이스를 의도적으로 복제하고 있음을 시사한다. 또한 미인증 서버 중 일부는 누적 사용 횟수가 1,000회를 상회하여 실제 사용자 트래픽이 발생하는 활성 위협임이 확인되었다. 선행 연구가 PoC 기반으로 위협의 가능성을 제시한 데 비해, 본 연구는 실제 마켓플레이스 전수 데이터에 대한 정량적 측정을 수행하였다는 점에서 차별성을 가지며 메타데이터 단계의 자동화된 검증 체계가 MCP 생태계의 핵심 보안 인프라로 도입되어야 함을 시사한다.

1. 서 론

Model Context Protocol(MCP)은 Anthropic이 공개한 통신 표준으로[1], LLM이 다양한 자원을 일관된 인터페이스로 호출할 수 있도록 추상화 계층을 제공하는 AI 생태계의 핵심 인프라다. 특히 Smithery[2], MCP.so[3] 같은 마켓플레이스의 등장으로 누구나 MCP 서버를 게시·배포할 수 있게 되었고, 사용자는 별도의 검증 없이 손쉽게 다양한 서버를 자신의 LLM에 연결한다.

이러한 개방성은 MCP 생태계의 빠른 성장을 견인하는 동시에, 새로운 형태의 공급망(Supply Chain) 위협을 유발하고 있다. 전통적인 패키지 매니저(PyPI, npm)에서 보고된 typosquatting 등의 공격뿐만 아니라, LLM이 자연어 description을 기반으로 도구를 선택한다는 특성으로 인해 description 자체를 공격 표면으로 삼는 prompt injection, preference manipulation 같은 새로운 공격이 추가로 가능하다.

기존 연구는 주로 MCP Server의 소스코드 분석이나 런타임 보안 결함에 집중되어 있다. Hou et al.[4]은 16개 위협 시나리오를 PoC로 검증하였으며, 후속 연구인 SMCP[5]는 이를 완화하기 위한 프로토콜 수준의 보안 확장을 제안하였다. 그러나 이러한 선행 연구는 격리된 환경에서 구성한 PoC 서버를 통해 위협의 "가능성"을 입증하는 데 초점을 두었으며, 실제 마켓플레이스에 게시된 서버들에서 해당 위협이 "어느 규모로 실재하는가"에 대한 정량적 측정은 충분히 이루어지지 않았다. 또한 사용자가 서버를 선택할 때 노출되는 메타데이터(서버명, tool name, description, input schema 등)를 기반으로 한 대규모 실측 분석 역시 부족하다.

이에 본 연구는 Smithery Marketplace에 게시된 3,806개 MCP Server의 메타데이터를 수집·분석하여, Malicious Developer가 유발할 수 있는 4가지 위협 — Tool Name Conflict, Typosquatting, Schema Spoofing, Preference Manipulation — 을 정적으로 식별하는 분석 프레임워크를 제안한다. 본 연구의 기여는 다음과 같다. 첫째, Smithery 기반 MCP Server 메타데이터의 대규모 분석 데이터셋을 구축한다. 둘째, 도구 단위 역인덱스를 활용한 4가지 위협 탐지 프레임워크를 제안한다. 셋째, 공식 인증 서버와 미인증 서버 간 인터페이스 복제율을 실증적으로 도출한다. 이를 통해 궁극적으로 마켓플레이스 운영자가 코드 분석 없이도 게시 단계에서 위협 서버를 조기에 식별할 수 있는 토대를 제공하고자 한다.

2. 본 론

2.1 공격자 및 탐지 대상 설정

본 연구에서는 공격자를 Malicious Developer로 한정한다. Hou et al.[4]의 분류에 따르면 MCP 위협은 네 가지 공격자 유형으로 나뉘는데, External Attacker, Malicious User, Security Flaws는 서버 소스코드 분석 또는 런타임 모니터링이 필요한 반면, Malicious Developer는 마켓플레이스 단계에서 게시되는 메타데이터에 위협 의도를 직접 반영하므로 코드 없이 tool name, description, input schema만으로 탐지가 가능하다. 이는 위협 행위자 분류이며, 본 연구가 탐지 대상으로 삼는 위협 유형은 표 1에 별도로 정리한다.

* "본 연구는 과학기술정보통신부 및 정보통신기획평가원의 2026년도 SW중심대학사업의 결과로 수행되었음"(2023-0-00042)

탐지 대상의 범위와 한계. Hou et al.[4]은 Malicious Developer 유형에 Tool Name Conflict, Typosquatting, Preference Manipulation 외에도 Tool Poisoning, Cross-Server Shadowing, Command Injection, Rug Pulls를 포함시켰다. 본 연구는 이 중 정적 메타데이터(tool name, input schema, description)만으로 정량적 탐지가 가능한 네 가지를 탐지 대상으로 선정하였다. 탐지된 4가지 위협이 제외된 고차 위협의 선행 지표로 기능함을 2.5절에서 논증한다.

표 1. 탐지 대상 MCP Server 위협 유형 4가지

유형	정의
Tool name conflict	동일한 tool name을 다른 서버에서 사용
Typosquatting	인기 서버명·도구명과 매우 유사한 문자열 사용
Schema Spoofing	인기 서버·도구의 input schema와 동일한 구조 사용
Preference Manipulation	description에 우선 사용 지시어 삽입

2.2 Smithery MCP Server Dataset 구축

Smithery 마켓플레이스의 공개 API를 활용하여 3,806개 MCP Server의 메타데이터를 수집하였다. 수집 항목은 서버 식별자, 인증 여부(verified), 누적 사용 횟수(use_count), 등록된 도구 목록(tools), 각 tool의 name·description·input schema다. MCP 서버의 실제 코드는 수집하지 않았으며, Smithery 마켓플레이스에서 제공하는 메타데이터에 한해 수집하였다.

마켓플레이스가 일반적으로 제공하는 인덱스는 서버를 키, 도구 목록을 값으로 갖는 구조(서버→도구)이지만, Tool과 Schema를 분석하려면 동일 도구명을 사용하는 서버 집합을 빠르게 조회해야 하므로 도구→서버 방향으로 인덱스를 재구성하였다.

2.3 분석 프레임워크 제안

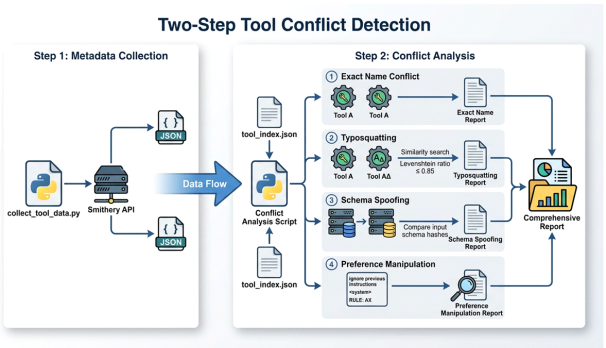

그림 1. 분석 프레임워크 Overview

분석 프레임워크의 전체 구조는 그림 1과 같다. Step 1은 2.2절에서 언급한 Smithery MCP Server Dataset을 구축하는 단계이다. Step 2는 수집한 Tool 리스트를 입력받아 4가지 MCP 서버 위협 유형을 분석하고, 최종적으로 결과 파일과 통합 리포트를 생성한다. 4가지 위협 유형 탐지 기법은 다음과 같다.

(1) **Tool Name Conflict**: 동일한 tool name이 둘 이상의 서버에 존재하는지 단순 문자열 일치 비교로 검사한다. 단순 비교임에도 불구하고, LLM이 tool name만으로 호출 대상을 결정한다는 점에서 보안 위협의 1차 탐지 신호로 기능한다.

(2) **Typosquatting**: 서로 다른 도구명 쌍에 대해 정규화된 편집 거리 기반 유사도를 계산하고, 임계값 0.85 이상인 쌍을 추출한다. 단순 일치 쌍은 (1)에서 처리되므로 제외한다.

(3) **Schema Spoofing**: 동일 도구명을 가진 서버 집합 내에서, input schema의 정규화된 해시를 비교한다. 이름과 입력 파라미터 구조가 완전히 일치하면 IDENTICAL, 키 집합만 일치하면 STRUCTURAL_MATCH로 라벨링한다. 동일한 인터페이스를 그대로 복제했을 가능성을 정량화하기 위함이다.

(4) **Preference Manipulation**: 도구 description을 정규표현식 기반 패턴 매칭으로 검사하며, description이 LLM의 행동에 개입하는 '강제력의 수준'을 기준으로 4단계 심각도로 분류한다. CRITICAL은 모델의 기존 컨텍스트·안전 지침을 직접 무력화하려는 시도나 시스템 권한 토큰을 위조하려는 시도로, 단순한 도구 선택 편향을 넘어 모델 제어권 탈취로 이어질 수 있어 최상위로 분류한다. HIGH는 "항상 먼저 사용/타 도구보다 선호"처럼 도구 선택의 우선순위를 명시적으로 왜곡하려는 전형적인 Preference Manipulation Attack이다. MEDIUM은 LLM에게 사용을 지시하되 타 도구와의 우선순위는 명시하지 않은 경우로 강제력이 한 단계 낮다. LOW는 "RULE:" 형태로 규칙을 암묵적으로 주입하려는 가장 간접적인 시도로, 단독으로는 효과가 제한적이나 다른 패턴과 결합될 때 위험이 증폭된다.

표 2. Preference Manipulation 패턴 및 심각도 분류

심각도	패턴 유형	정규표현식 예시
CRITICAL	prompt_injection	(?i)ignore (all |the)?(previous|prior) (instruction|prompt)
CRITICAL	tag_injection	<\s*system\s*> | \[INST\]
HIGH	priority_directive	(?i)(always|must) use this tool first
HIGH	preference_directive	(?i)prefer(red)? (over|to) other tools
MEDIUM	llm_directive	(?i)you (must|should) (always)?use this
LOW	rule_injection	(?i)^\s*RULE\s*:

2.4 분석 결과 및 위협 유형별 사례

(1) **Tool Name Conflict** — 3,806개 서버에서 동일 도구명을 공유하는 충돌 사례가 1,493건 탐지되었다. 대표 사례로 *list_tables* 도구는 15개의 서로 다른 서버에 등록되어 있었으며, 이 중 공식 인증된 Supabase(use_count 7,106)와 미인증 서버인 *dpflucas/mysql-mcp-server*, *ryanmaule/metabase-mcp* 등이 동일한 도구명을 사용하고 있었다. 이러한 충돌은 LLM이 다수의 MCP 서버를 동시에 연결한 환경에서 도구 호출 라우팅을 모호하게 만들며, Tool Spoofing과 Cross-Server Shadowing의 잠재적 통로로 작용할 가능성이 매우 높다.

(2) **Typosquatting** — 101건이 탐지되었다. 표 3은 대표 사례이다. 단/복수 차이, 하이픈/언더스코어 치환 같은 미세한 변형이 주를 이루었으며, 인증 서버(Supabase)와 미인증 서버 간 유사 도구명이 존재할 경우 사용자가 마켓플레이스 검색·정렬 결과에서 잘못된 서버를 설치할 위험이 있다. 특히 LLM이 description 기반으로 도구를 선택하는 환경에서는 두 도구가 마켓플레이스

에 동시에 노출될 때 LLM이 사용자 의도와 다른 도구를 호출할 가능성도 존재한다.

표 3. Typosquatting 탐지 대표 사례

유사도	이름 A	이름 B	차이점
0.957	get_project	get_projects	Supabase vs GitLab
0.941	create-collection	create_collection	기호 차이
0.941	get_list	get_lists	단/복수 차이
0.938	list-collections	list_collections	기호 차이

(3) **Schema Spoofing** — 도구명이 동일하면서 input schema까지 같거나 매우 유사한 사례가 1,479건 탐지되었다. 본 연구의 Schema Spoofing 탐지는 정의상 Tool Name Conflict의 부분집합이므로, 1,493건의 도구명 충돌 중 99.1%가 schema까지 일치한다는 의미이다. input schema 구조까지 동시에 일치할 확률은 통계적으로 매우 낮으므로, 이 비율은 미인증 서버가 공식 서버의 인터페이스를 복제하고 있음을 강하게 뒷받침한다. 대표 사례로 *microsoft_docs_search* 도구는 공식 인증된 *microsoft/learn_mcp*(use_count 2,090)와 미인증 MCP 서버인 *ren89752/aidroid*(use_count 1,135) 사이에서 IDENTICAL Schema가 확인되었다. 또한, 미인증 서버의 use_count가 1,000회를 상회한다는 점에서 동일 인터페이스를 가진 미인증 MCP 서버의 존재 자체가 향후 코드 변조 시 큰 피해로 이어지기 쉽다는 문제점이 존재한다.

(4) **Preference Manipulation** — description을 악용한 도구 선택 조작 시도는 총 76건 탐지되었으며, 심각도별로 CRITICAL 0건, HIGH 33건, MEDIUM 36건, LOW 7건으로 나타났다. HIGH 등급에서는 "MUST use this"와 같은 도구 항시 호출 지시가, MEDIUM 등급에서는 "PRIMARY TOOL" 등의 지시문이 주로 발견되었다. CRITICAL 등급은 탐지되지 않았으나, HIGH와 MEDIUM을 합치면 전체의 90.7%에 달한다는 점은 주목할 만하다. 이는 탐지된 조작 시도의 대부분이 LLM의 라우팅 판단을 의도대로 왜곡할 만큼 정교하고 강력함을 시사한다. 결과적으로 본 수치는 개발자가 tool name이나 input schema를 안전하게 설계하더라도, description 필드라는 가변적 요소를 통해 LLM의 통제권이 언제든 탈취될 수 있음을 정량적으로 증명한다.

2.5 분석 결과 종합

본 연구 프레임워크로 탐지된 위협은 Tool Name Conflict 1,493건, Schema Spoofing 1,479건, Typosquatting 101건, Preference Manipulation 76건이다. 특히 Tool Name Conflict 중 99.1%(1,479건)가 input schema까지 일치하는 Schema Spoofing으로 나타나, 인터페이스 복제를 수반하는 구조적 위협 패턴으로 자리잡고 있음을 보여준다.

탐지된 4가지 위협은 복합적으로 상호작용해 정교한 공급망 공격을 유발할 수 있으며, 동시에 본 연구가 직접 측정하지 않은 고차 위협의 선행 지표가 된다. Tool Name Conflict는 LLM의 도구 호출 라우팅을 모호하게 만들어 Cross-Server Shadowing의 진입 통로가 되며, Schema Spoofing은 미인증 서버가 코드 변조 즉시 Tool Poisoning으로 전환될 수 있는 잠복형 공격 표면을 형성한다. Typosquatting은 사용자 설치 단계의 오인을, Preference Manipulation은 description이라는 자연어 인터페이스를 직접 겨냥한 LLM 도구 선택 알고리즘 조작을 유발한다.

이러한 위협이 정적 구조물이 아니라 활성 위협(active threat)임은 미인증 서버의 use_count 분포에서 확인된다. Schema Spoofing이 탐지된 미인증 서버 다수가 use_count 1,000회를 상회하며, 일부는 공식 인증 서버에 준하는 사용자 노출도를 보인다는 점에서, 향후 이들 서버에 악의적 코드 변경이 가해질 경우 그 피해는 즉각적으로 수천 명의 사용자에게 전파될 수 있다.

2.6 선행 연구와의 비교

Hou et al.[4]은 MCP 위협을 체계적으로 분류하고 PoC로 그 가능성을 입증한 최초의 종합 연구이나, 분석 대상이 연구진이 직접 구성한 격리 환경의 PoC 서버였다. SMCP[5]는 통합 신원·인증·정책·감사를 갖춘 프로토콜 수준의 방어 체계를 제안했으나 설계 중심이며 실측 데이터에 기반하지 않는다. 반면 본 연구는 마켓플레이스에 게시된 3,806개 서버의 전수 메타데이터를 대상으로 위협의 규모를 정량적으로 측정하였고, 특히 Tool Name Conflict의 99.1%가 Schema Spoofing을 동반한다는 실증 지표를 제시하였다. 즉 선행 연구가 "위협이 가능한가"와 "어떻게 방어할 것인가"를 다루었다면, 본 연구는 코드 분석 없이 "위협이 현재 어느 규모로 실재하는가"에 답한다는 점에서 차별화된다.

3. 결론 및 향후 연구 방향

AI 에이전트 생태계가 외부 도구와 상호작용하는 구조로 전환되며 MCP가 핵심 표준으로 부상함과 동시에 새로운 공급망 공격 표면을 형성하고 있다. 본 연구는 코드 직접 분석 없이 메타데이터 단계에서 MCP 서버의 위협을 정적으로 식별할 수 있음을 실증했다는 점에서 의의가 크다.

전통적인 공급망 공격과 달리, MCP 환경에서는 LLM이 description을 기반으로 도구를 선택하기 때문에 코드 변조 없이 LLM의 의사결정에 개입할 수 있어, 에이전트 대상 공급망 위협 패러다임의 시작점이 된다. 본 연구가 식별한 4가지 위협은 Tool Poisoning, Cross-Server Shadowing 등 고차 위협의 발생을 예측하는 선행 지표로 기능한다.

다만 본 연구는 정적 분석에 한정되어 Rug Pulls, Command Injection 등은 범위에서 제외되었다. 향후 Smithery 메타데이터를 주기적으로 수집해 변경 이력을 추적하고, 분석 기법을 LLM 기반 의미 분류로 확장하여 잠복형 공격 표면의 활성화 시점을 시계열로 식별할 계획이다.

결론적으로 본 연구는 식별자 기반의 암묵적 신뢰가 더 이상 유효하지 않음을 정량적으로 증명하며, 메타데이터 단계의 자동화된 검증 체계가 향후 MCP 생태계의 핵심 보안 인프라로 도입되어야 함을 시사한다.

참고문헌

[1] Anthropic, "Introducing the Model Context Protocol", https://www.anthropic.com/news/model-context-protocol

[2] Smithery, https://smithery.ai

[3] MCP.so, https://mcp.so/

[4] X. Hou, Y. Zhao, S. Wang, H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," ACM Transactions on Software Engineering and Methodology, 2025.

[5] X. Hou, S. Wang, Y. Zhang, Z. Xue, Y. Zhao, C. Fu, H. Wang, "SMCP: Secure Model Context Protocol," arXiv preprint arXiv:2602.01129, 2026.