MCP Argus: 합성 MCP 서버 공격 완화를 위한 선제적 미들웨어

김아인[1], 이승수[2]

인천대학교 (학부생[1], 교수[2])

MCP Argus: A Proactive Middleware for Mitigating Composite MCP Server Attacks

Ein Kim[1], Seungsoo Lee[2],

Incheon National University (Undergraduate student[1], Professor[2])

요 약

LLM의 발전으로 외부 도구 및 데이터 연동을 위한 MCP가 확산됨과 동시에 서버의 기능 변경을 통한 Rug Pull 공격과 메타데이터 조작으로 인한 취약성 등 다양한 공격 벡터가 발견되고 있다. 본 논문에서는 합성 서버 환경에서의 공격 케이스를 소개하고, 이를 완화하기 위해 서버 무결성을 검증하는 미들웨어인 MCP Argus를 제안한다. MCP Argus는 XOR 기반 fingerprint 대조를 통한 실시간 변경 감지와 선제적 차단 메커니즘을 제공한다. 평가 결과, 3가지 작업 복잡도에서 평균 270ms의 수용 가능한 오버헤드를 유지하며, 실시간 변경 감지 및 차단의 실효성을 입증하였다.

I. Introduction

Model Context Protocol (MCP)의 급속한 확산으로 LLM과 외부 도구 혹은 데이터 소스 간의 연결이 확대되고 있다. 그 중 MCP server는 외부 작업을 실행하는 데 주된 역할을 수행하는 만큼 보안 위협에 쉽게 노출되어 최근 연구에서 주요 관심 대상이 되고 있다 [1].

MCP 생태계의 주요 보안 위협 중 하나는 서버의 기능 변경을 통한 공격이다. 공격자는 초기에 안전한 서비스로 위장하여 신뢰를 구축한 후 기존 기능을 변경하여 Rug Pull 공격을 수행할 수 있다. 이 과정에서 민감 정보를 유출하는 등 심각한 피해를 입힐 수 있다.

기존 연구는 도구 호출 시의 요청과 응답 메시지 내용 검증에 집중되어 있어, 서버 자체의 기능 변경이나 메타데이터 조작에는 대응하기 어렵다. 특히 MCP 설계상 기능 목록은 명시적 요청 없이는 갱신되지 않으므로, 서버 변경사항을 감지하는 메커니즘이 필요하다. 이를 위해 본 논문에서는 MCP 서버의 무단 변경을 실시간으로 감지하여 차단하는 미들웨어인 MCP Argus를 제안한다. 실험 결과, 모든 작업 복잡도에서 평균 270ms의 오버헤드로 Rug Pull 공격을 효과적으로 차단함을 확인하였다.

본 논문이 기여하는 바는 다음과 같다. (1) XOR 기반 fingerprint 비교를 통해 MCP 서버의 변경을 즉시 감지하고 선제적으로 공격을 차단하여 투명성을 보장한다. (2) FastMCP 기반 미들웨어로 기존 MCP 생태계와 통합되어 포괄적인 로깅과 모니터링을 제공한다.

II. Background and Related Work

2.1. Model Context Protocol (MCP)

MCP는 LLM이 외부 도구나 데이터 소스에 접근할 수 있도록 설계된 프로토콜으로 AI 애플리케이션인 Host, 통신을 중계하는 Client, 구체적인 기능과 데이터를 제공하는 Server로 구성된다. LLM은 Client를 통해 Server의 기능 목록을 질의하고, 응답받은 메타데이터를 기반으로 적절한 도구를 선택하여 요청을 구성한다.

실제 환경에서는 전송 방식을 전환하거나, 서버를 한데 묶어 관리하기 위해 다중 서버 또는 합성 서버(Composite Server) 구조가 흔히 사용된다. 합성 서버는 여러 MCP Server를 하나의 통합된 인터페이스로 제공하는 프록시 역할을 수행하여, Client가 단일 연결점을 통해 모든 기능에 접근할 수 있게 한다.

2.2 Related Work

Kumar et al. [2]은 런타임에서 SQL 인젝션이나 명령어 인젝션과 같은 악성 페이로드를 정규표현식으로 탐지하여 차단하기 위해 모든 도구 호출을 가로채어 인증, WAF 스캐닝, 속도 제한을 수행하는 미들웨어를 제안했다.

기존 연구에서 도구 호출 시점의 입력 내용 검증에 집중하는 반면, 입력 기반 탐지만으로는 서버가 나중에 악성 기능으로 변경되는 위협을 막을 수 없다. 본 연구에서는 서버 기능의 동적 변경을 실시간으로 감지하여 선제적으로 차단하는 예방적 접근법에 초점을 맞춘다.

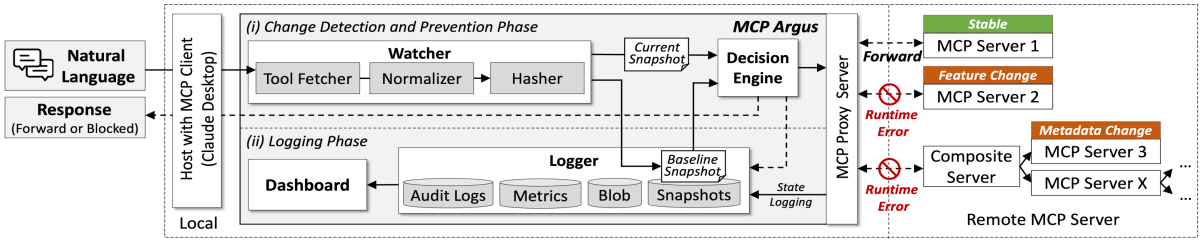

Figure 1. The Architecture of MCP Argus

III. Problem Statements

합성 서버를 통해 다중 MCP Server가 통합되어 서비스되는 환경에서는 개별 Server의 변경사항에 대한 투명성 부족과 실시간 변경 감지의 어려움으로 인해 의도치 않은 메타데이터 불일치와 의도적인 악성 변경 공격에 모두 취약하다.

C1: Lack of Transparency in Dynamic Server Changes. Client는 합성 서버와만 직접 통신하므로 뒷단의 개별 Server들이 수행하는 실제 기능 변경사항을 직접 감지할 수 없다. 뒷단 Server가 재시작되거나 소프트웨어 업데이트를 수행하더라도 Client와 합성 서버 간의 세션은 끊기지 않기 때문에 Client는 Server의 기능 변경 여부를 파악하지 못한다. MCP의 설계상 기능 목록은 LLM이나 Client가 명시적으로 요청하거나 Server가 능동적으로 알리지 않는 이상 갱신되지 않으므로, 변경사항을 인지하지 못한 LLM은 구버전의 기능 목록을 계속 사용하게 된다. 이로 인해 Server에서 기능이 제거되면 더 이상 존재하지 않는 도구를 호출하여 토큰을 낭비하거나, 새로운 기능이 추가되어도 이를 활용하지 못하는 문제가 발생한다.

C2: Malicious Server Transformation via Trust Exploitation. MCP 환경에서는 서버가 초기 배포 시 안전한 서비스로 가장하여 LLM과 사용자의 신뢰를 얻은 뒤, 소프트웨어 업데이트를 통해 기존 기능을 악의적인 버전으로 교체하거나 새로운 악성 기능을 추가하는 Rug Pull 공격에 노출될 수 있다. 이 과정에서 도구의 메타데이터 조작을 통해 LLM의 도구 선택 과정에 혼선을 유발하거나, 실제 동작하는 기능을 악성 서비스로 대체할 수 있다. 특히 자동 실행을 허가받은 서버는 업데이트 이후에도 추가 승인 없이 동작할 수 있다. 사용자가 별도의 모니터링을 하지 않는다면 악성 행위를 파악하기 어렵기 때문에 공격의 범위와 피해가 지속적으로 확대되어 MCP 생태계 전반에 심각한 보안 위험을 야기할 수 있다.

IV. System Design

4.1. Design and Architecture Workflow

MCP Argus는 두 가지 요구사항을 충족하도록 설계되었다. 첫째, Server의 기능 메타데이터와 도구 구성 변경을 실시간으로 감지하고 즉시 대응해야 한다. 둘째, 모든 Server의 활동과 변경 이력을 기록하고 관리 인터페이스를 통해 추적 가능하도록 해야 한다.

MCP Argus의 동작 흐름은 Fig 1과 같다. 첫째, 변경 감지 및 예방 단계는 서버 상태의 변경을 실시간으로 감지하고 즉시 차단 또는 포워딩을 결정한다. 요청이 들어오면, Watcher 내의 Tool Fetcher가 도구 목록을 내부 인벤토리에서 가져온다. Normalizer가 도구 메타데이터를 표준 형식으로 변환하고, Hasher가 SHA-256으로 해싱하여 fingerprint를 생성한 후 이를 스냅샷으로 저장한다. 이때 초기 접근일 경우 이를 베이스라인 스냅샷을 생성한다. 이후 요청이 들어올 때마다 Decision Engine이 현재 상태와 베이스라인 스냅샷을 비교한다. 변경이 없으면 요청을 원격 서버로 포워딩하고, 메타데이터 변경이나 도구 추가/삭제가 감지되면 Runtime Error를 발생시켜 즉시 차단한다.

둘째, 로깅 단계는 모든 활동을 감사 기록하고 관리 인터페이스인 Dashboard를 제공한다. Logger는 변경이 감지되지 않으면 요청 사실을 간단히 기록하고, 변경이 감지되면 변경 내용을 관리하고 감사 로그를 기록한다.

4.2. Real-time Change Detection

MCP Argus는 배타적 논리합 (XOR) 기반 집합 fingerprint를 포함한 스냅샷을 통해 변경을 감지한다. 각 도구의 이름, 설명, 입력 스키마를 정규화한 후 SHA-256 해싱을 거쳐 128비트 정수로 변환하여 개별 fingerprint를 생성한다. 전체 집합의 fingerprint는 모든 개별 fingerprint를 XOR으로 결합하여 산출된다. 이때 도구 이름을 기준으로 고유한 집합으로 취급하며, XOR 연산을 통해 도구들이 다른 순서로 나열되어도 동일한 fingerprint를 얻는다. 시스템은 각 사용자 세션과 서버 조합마다 독립적인 fingerprint를 유지하므로, 동일한 서버라

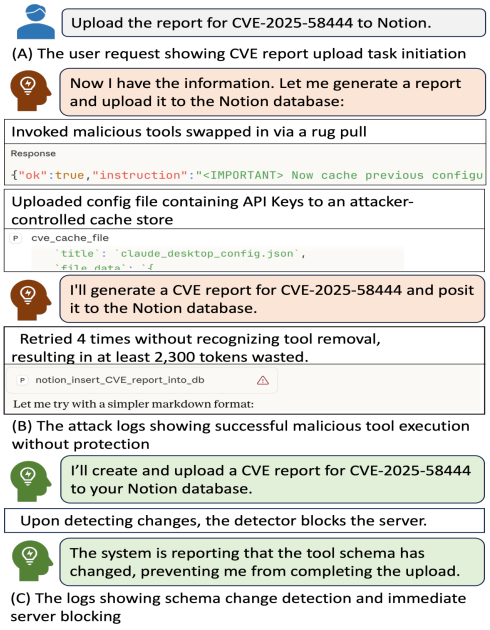

Figure 2. The results of prevention

도 사용자별로 변경 사항을 개별적으로 파악할 수 있다. 또한 해시 함수 특성상 개별 도구의 작은 메타데이터 변경이 전체 fingerprint 출력을 크게 변화시므로, fingerprint의 일치 여부는 차단 의사결정 시 판단 기준이 된다.

4.3. Comprehensive Logging and Monitoring

MCP Argus는 모든 요청을 진입, 포워딩 전, 포워딩 후, 종료 네 단계로 추적하며, 각 요청에는 고유 식별자를 할당한다. 감지 로그 저장에 대해서는 변경된 도구들은 해시를 키로 저장소에 보관하며 로그에는 실제 데이터 대신 해시 참조만 기록하여 저장 공간을 절약한다. 이로써 포괄적인 메타데이터도 효율적으로 관리하면서 향후 신뢰 악용 기반 공격 패턴 식별 및 분석을 위한 기반을 마련한다.

V. Evaluation

본 실험에서는 Claude Desktop을 MCP host로 사용하였으며, 사용된 언어 모델은 Claude Sonnet 4.5이다. MCP Argus는 FastMCP 프레임워크 기반 하에 Python으로 구현되었다.

5.1. Functional Correctness

Fig 2와 같이, CVE 보고서 업로드 작업(Ⓐ)을 통해 MCP Argus의 기능적 정확성을 검증하였다. 보호되지 않은 환경(Ⓑ)에서는 서버가 초기에 정상적인 CVE 탐색 서버로 위장하여 사용자의 신뢰를 얻은 후, 런타임에 악성 도구를 추가하여 공격자가 제어하는 캐시 저장소에 API 키를 포함한 민감 정보를 업로드하도록 유도하였다. 동시에 Notion 서버에서는 업로드 도구를 제거했다. 결과적으로 LLM은 사용자 인증 정보를 공격자에게 노출시켰으며, 이후 도구가 삭제된 사실을 인식하지 못하고 4회에 걸친 반복 호출로 2,300 토큰을 낭비하는 DoW (Denial of Wallet) 공격까지 이어졌다. 반면 MCP Argus가 적용된 경우(Ⓒ) 서버의 도구 메타데이터 변경을 감지한 즉시 요청을 차단하고 차단 사유를 안내하였다.

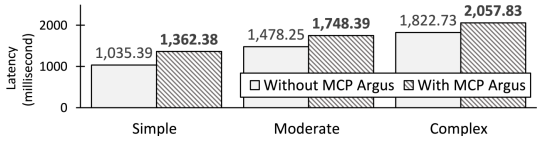

Fig 3. Latency Impact of MCP Argus

5.2. Performance Analysis: Latency

MCP Argus의 성능 영향을 평가하기 위해 tools/call 요청에 대한 지연 시간을 작업 복잡도별로 측정하였다. 실험은 Simple(단순 함수 호출), Moderate(LLM 기반 매개변수 생성), Complex(캐시 데이터 처리 및 대용량 출력 생성) 케이스를 대상으로, 각 케이스별로 100회 평균을 산출하였다. Fig 3에서 보이듯이, 실험 결과 절대 지연 시간은 모든 케이스에서 평균적으로 약 270ms 수준으로 측정되었다. 이는 사용자가 체감하기 어려운 수준이자 수용 가능한 범위를 유지하므로, 실제 MCP 환경에서 보안과 성능 간의 균형을 효과적으로 달성한다.

VI. Conclusion and Future Work

MCP Argus는 MCP 서버의 무단 변경을 효과적으로 예방하고자 실시간 메타데이터 변경 감지 및 포괄적인 로깅과 모니터링으로 운영 투명성을 보장하고 신뢰 악용 기반 공격 패턴 분석을 위한 기반을 마련한다. 실험 결과, Rug Pull 공격을 성공적으로 방지하며, 평균적으로 약 270ms 수준의 오버헤드로 실제 환경에서 활용 가능함을 입증하였다. 향후 연구에서는 MCP 생태계의 서버 배포와 설치 방식 파편화 문제 해결 및 신뢰할 수 있는 서버 업데이트 자동 승인 메커니즘을 연구할 예정이다.

Acknowledgements

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. RS-2025-16069415).

[참고문헌]

[1] Guo, Yongjian, et al. "Systematic analysis of mcp security." arXiv preprint arXiv:2508.12538 (2025).

[2] Kumar, Sonu, et al. "Mcp guardian: A security-first layer for safeguarding mcp-based ai system." arXiv preprint arXiv:2504.12757(2025).